BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 468 (January 2024)

VLSI Circuits and Design Laboratory

Final Project Report

Section: G1 Group: 07

16 bit Serial in Serial out FIFO Shift Register

Course Instructors:

Nafis Sadik, Lecturer
Rafid Hassan Palash, Part-Time Lecturer

Signature of Instructor: ______________________________

Academic Honesty Statement:

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: **Full Name:** **Student ID:**	Md Jarjis Mondal 1806068	**Signature:** **Full Name:** **Student ID:**	Mehedi Hasan 1906075
Signature: **Full Name:** **Student ID:**	Md Faiyaz Abid 1906079	**Signature:** **Full Name:** **Student ID:**	Junaidul Islam Sikder 1906041

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. ***You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.***

Table of Contents

1 Absrtact:

This project focuses on the design and implementation of a 16-bit Serial-In Serial-Out shift register capable of both left and right shift operations. The design was implemented in Verilog and verified using a combination of directed and layered verification methodologies to ensure comprehensive testing. Subsequently, the design was optimized for timing constraints, including clock frequency, input delay, and output delay. Synthesis was performed using Cadence tools, resulting in a netlist that was then used for physical design and layout. The physical design process involved floorplanning, placement and routing, and post-route timing analysis. Finally, rigorous Design Rule Checks (DRCs) were conducted to ensure the manufacturability of the design. The project successfully demonstrated the design and implementation of a functional and optimized 16-bit SISO shift register.

2 Introduction:

A Serial-In Serial-Out shift register is a sequential logic circuit that shifts data in and out one bit at a time in a serial manner. It is made up of a chain of flip-flops connected in series.In our project, 16 bit data is taken as input and with every clock pulse, data is shifted serially. Our designed resgister can perform both left and right shifting. We can load the value anytime we want. We have implemented the code both in EdaPlayground and Cadence. Directed verification is done to check whether our code results desired output or not. But in order to verify for all possible cases, we verified in a structured way by layered verification. Next, we optimized time constrains such as clock frequency, input delay and output delay. Then we have synthesized our design using the optimized time constrains. Synthesis was done in cadence and after that it generated a synth.v file. This file is used in physical designing. Physical designing was done in Cadence Innovus. We selected suitable die size for our design. Finally, we cross checked our DRC results for any errors. DRC rsults were compiled by Cadence Virtuoso. In this way, we completed our project successfully.

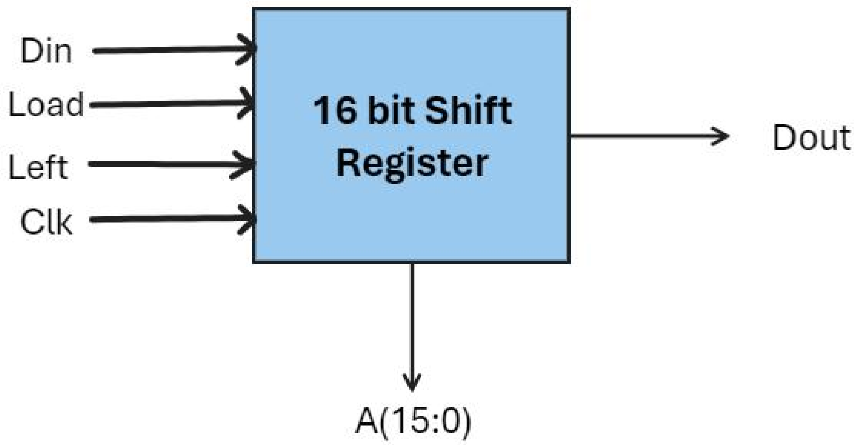
3 Specifications:

- 16 bits are allocated for parallel load, A(15:0)
- 1 bit serial input, Din.
- 1bit serial output, Dout.
- Clock is used to shift the data in a synchronized way.
- Operations:

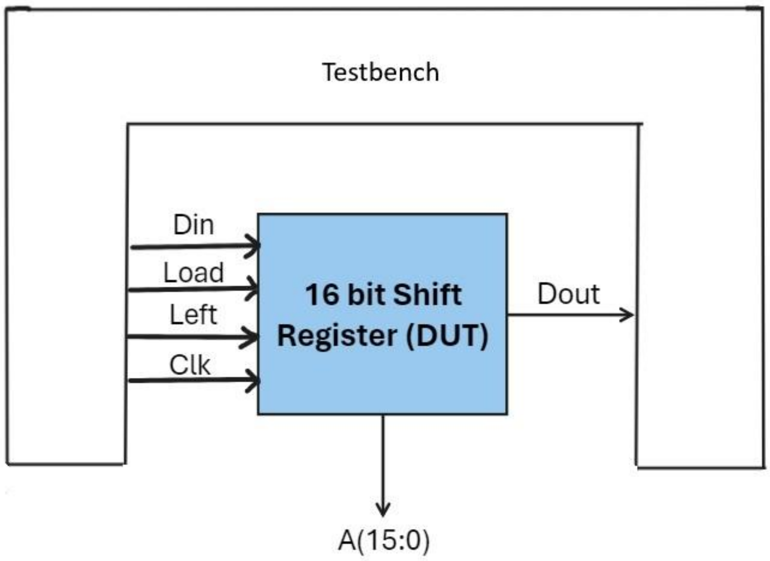
Left	Operation
0	Shift right
1	Shift left

Load	Operation
0	Shift data
1	Load the data

4 Block diagram:



Here, this is a simple illustration on our 16 bit shift register. All the signals are synchronised, so we have a clock for that. Left and Load are operational single control bits. And Din is the input single bit here. Due to shifting everytime, Dout, a single bit is also getting out of the register. Finally a 16 bit parallel data is loaded.



This is the illustration for our register along with test bench. The Device Under Test (DUT) is connected to correspoding ports. It checks the result with the expected ourtput and generate a feedback.

5 RTL Code:

```
// 16bit Serial In Serial Out reg

module srp (clk,Load,Din,Left,Dout,A); //srp=shift registrer project
   input  clk;          // Clock
   input  Load;         //load
   input  Din;
   input [15:0] A;      // parallel load
   input  Left;         // (1: left,0:right)
   output reg Dout;     // Serial output
   reg [15:0] shift_reg; // 16-bit shift register


   always @(posedge clk)
    begin
     if (Load)
      begin
        shift_reg <= A; // parallel load
      end
     else if (Left)
     begin
        // Left shift
      Dout <= shift_reg[15];
      shift_reg <= {shift_reg[14:0], Din};
```

```
      end
      else
        begin
          // Right shift
          Dout <= shift_reg[0];
          shift_reg <= {Din, shift_reg[15:1]};

      end
  end
endmodule
```

Explanation:

This Verilog code defines a 16-bit Serial-In Serial-Out shift register module. It takes a clock signal, a load signal, a serial input, a parallel input, and a shift direction signal as inputs. It outputs a serial output. The module has a 16-bit shift register to store the data. When the load signal is high, the parallel input is loaded into the shift register. Otherwise, the data is shifted left or right based on the shift direction signal. The shifted-out bit is assigned to the serial output.

6 Directed Verification:

```
// Code your testbench here
// or browse Examples
module srp_tb;

  // Testbench signals
  reg clk;              // Clock signal
  reg Load;             // Load signal (parallel load of data)
  reg Din;              // Serial input data
  reg [15:0] A;         // Parallel load input
  reg Left;       // Shift direction (1: left, 0: right)
  wire Dout;            // Serial output

  // Instantiate the FIFO Shift Register (DUT)
  srp DUT (
      .clk(clk),
      .Load(Load),
      .Din(Din),
      .A(A),
      .Left(Left),
      .Dout(Dout)
  );
```

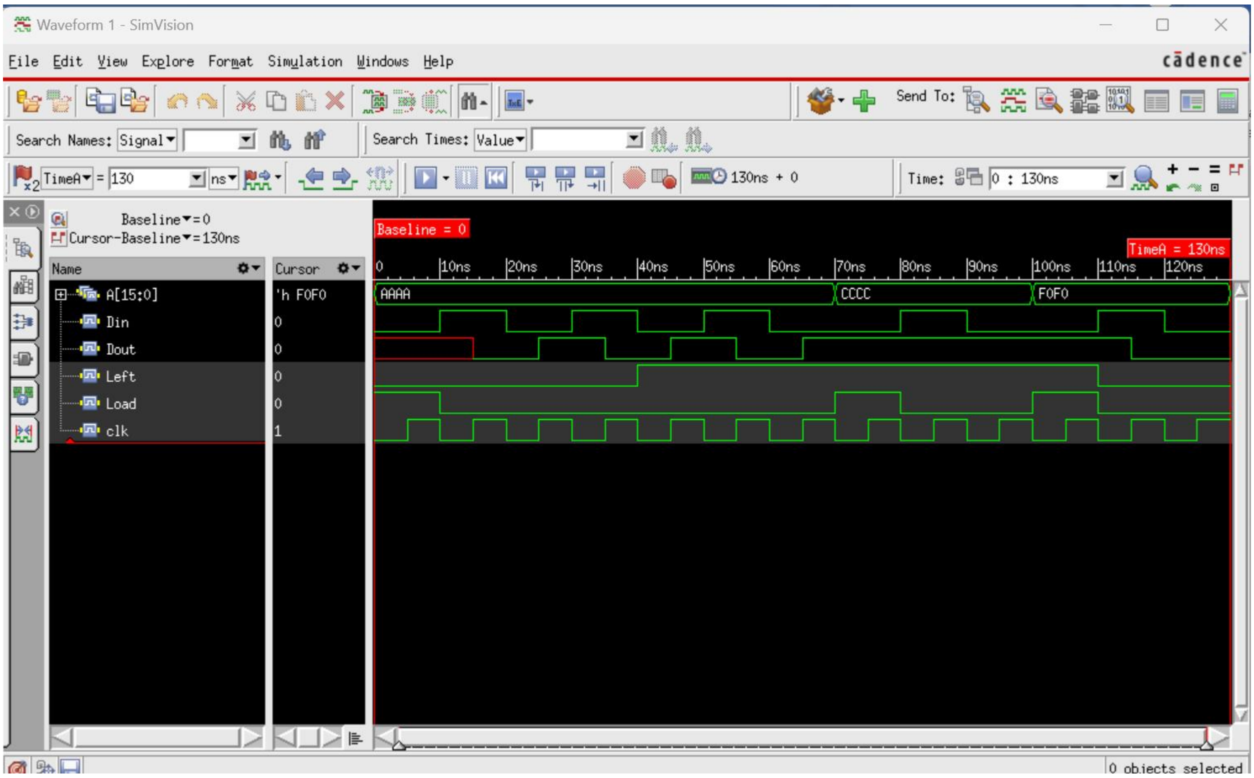
```
// Clock generation (period = 10 time units)
always begin
   #5 clk = ~clk;  // 100 MHz clock
end

// Test sequence
initial begin
   // Initialize signals
   clk = 0;
   Load = 0;
   Din = 0;
   A = 16'b0;
   Left = 0;

   // Apply reset (load = 1)
   Load = 1;
   A = 16'b1010101010101010; // Parallel load pattern
   #10; // Wait for one clock cycle

   Load = 0; // Disable load after initial parallel load
   // Shift right operation (shift_dir = 0)
   Left = 0;
   Din = 1; // Shift in data 1
   #10;
   Din = 0; // Shift in data 0
   #10;
   Din = 1; // Shift in data 1
   #10;

   // Shift left operation (shift_dir = 1)
   Left = 1;
   Din = 0; // Shift in data 0
   #10;
   Din = 1; // Shift in data 1
   #10;
   Din = 0; // Shift in data 0
   #10;

   // Test parallel load again
   Load = 1;
   A = 16'b1100110011001100; // New parallel load pattern
   #10; // Wait for one clock cycle
   Load = 0; // Disable load

   // Continue shifting left
   Left = 1;
   Din = 1; // Shift in data 1
   #10;
   Din = 0; // Shift in data 0
   #10;
```

```
        // Test with reset (parallel load) again
        Load = 1;
        A = 16'b1111000011110000; // New pattern
        #10;
        Load = 0;

        // Shift right operation again
        Left = 0;
        Din = 1;
        #10;
        Din = 0;
        #10;

        // Finish simulation
        $finish;
    end

    // Monitor the outputs
    initial begin
      $monitor("Time=%0t | Load=%b | Din=%b | Left=%b | Dout=%b | register=%b",
               $time, Load, Din, Left, Dout, DUT.shift_reg);
    end
  initial begin
 $dumpfile("dump.vcd");
 $dumpvars;
    #300;
    $finish;
 end

endmodule
```

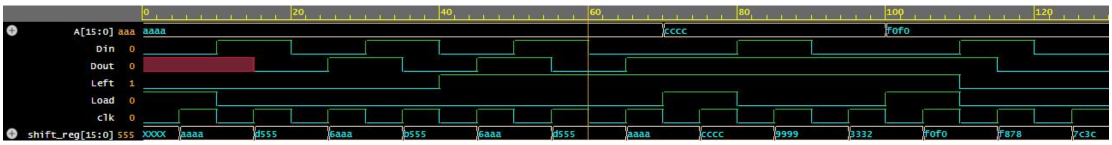
Explanation:

Here, the testbench instantiates the DUT, generates clock signals, applies various input stimuli (parallel load, serial input, shift direction), and monitors the output. The testbench includes a sequence of test cases to verify different functionalities of the shift register, such as parallel loading, left shift, right shift, and combinations of these operations. It also has a $monitor system task to display the current simulation time, input signals, output signal, and the internal state of the shift register. Additionally, it enables waveform dumping for detailed analysis.

Waveform (Using Cadence NCsim):



<u>Waveform (Using EdaPlayground):</u>



<u>Output:</u>

```
xcelium> run
Time=0 | Load=1 | Din=0 | Left=0 | Dout=x | register=xxxxxxxxxxxxxxxx
Time=5 | Load=1 | Din=0 | Left=0 | Dout=x | register=1010101010101010
Time=10 | Load=0 | Din=1 | Left=0 | Dout=x | register=1010101010101010
Time=15 | Load=0 | Din=1 | Left=0 | Dout=0 | register=1101010101010101
Time=20 | Load=0 | Din=0 | Left=0 | Dout=0 | register=1101010101010101
Time=25 | Load=0 | Din=0 | Left=0 | Dout=1 | register=0110101010101010
Time=30 | Load=0 | Din=1 | Left=0 | Dout=1 | register=0110101010101010
Time=35 | Load=0 | Din=1 | Left=0 | Dout=0 | register=1011010101010101
Time=40 | Load=0 | Din=0 | Left=1 | Dout=0 | register=1011010101010101
Time=45 | Load=0 | Din=0 | Left=1 | Dout=1 | register=0110101010101010
Time=50 | Load=0 | Din=1 | Left=1 | Dout=1 | register=0110101010101010
Time=55 | Load=0 | Din=1 | Left=1 | Dout=0 | register=1101010101010101
Time=60 | Load=0 | Din=0 | Left=1 | Dout=0 | register=1101010101010101
Time=65 | Load=0 | Din=0 | Left=1 | Dout=1 | register=1010101010101010
Time=70 | Load=1 | Din=0 | Left=1 | Dout=1 | register=1010101010101010
Time=75 | Load=1 | Din=0 | Left=1 | Dout=1 | register=1100110011001100
Time=80 | Load=0 | Din=1 | Left=1 | Dout=1 | register=1100110011001100
Time=85 | Load=0 | Din=1 | Left=1 | Dout=1 | register=1001100110011001
Time=90 | Load=0 | Din=0 | Left=1 | Dout=1 | register=1001100110011001
Time=95 | Load=0 | Din=0 | Left=1 | Dout=1 | register=0011001100110010
Time=100 | Load=1 | Din=0 | Left=1 | Dout=1 | register=0011001100110010
Time=105 | Load=1 | Din=0 | Left=1 | Dout=1 | register=1111000011110000
Time=110 | Load=0 | Din=1 | Left=0 | Dout=1 | register=1111000011110000
Time=115 | Load=0 | Din=1 | Left=0 | Dout=0 | register=1111100001111000
Time=120 | Load=0 | Din=0 | Left=0 | Dout=0 | register=1111100001111000
Time=125 | Load=0 | Din=0 | Left=0 | Dout=0 | register=0111110000111100
Simulation complete via $finish(1) at time 130 NS + 0
./testbench.sv:88         $finish;
xcelium> exit
```

We can verifiy our code from the plot or from the output. From the output, at t=15ns, Load=0, which means shifting operation is going on. And Left=0, which indicates right shifting operation. Input bit Din=1. So by checking the register value, we can verify that its right shifting and Dout is 0. We can also verify in the similar manner from the waveform as well.

7 Layered Verification

Testbench.sv file

```
// Code your testbench here
// or browse Examples
`include "testcase01.sv"
//`include "test.sv"
`include "interface.sv"

module srp_tb;
  bit clk;
  bit Load;
  bit Din;
  bit [15:0] A;
  bit Left;
```

```
  bit Dout;

  initial begin
    forever #5 clk =~clk;
  end

  int count=500;
  srp_if srpif(clk);

  test test01(count,srpif);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    #70000000;
    $finish;
  end

  srp DUT (
    .Load(srpif.Load),
    .Din(srpif.Din),
    .A(srpif.A),
    .Left(srpif.Left),
    .Dout(srpif.Dout),
    .clk(clk)
  );


endmodule
```

Explanation:

This Verilog code sets up a layered testbench for a 16-bit Serial-In, Serial-Out Shift Register module named srp. It includes a clock generation module, a stimulus generator, and a response checker. The testbench instantiates the srp module, applies various input stimuli, and verifies the output against expected values. The simulation is configured to run for a long duration, allowing for extensive testing.

Testcase01.sv file

```
`include "environment.sv"

program test(input int count, srp_if srpif);
  environment env;
```

```
  class testcase01 extends transaction;
    constraint c_s {
      //s inside {[0:1], [14:15]};
      Load inside {[0:1]};
      Din inside {[0:1]};
      Left inside {[0:1]};
      A inside {[0:65535]};
    }
  endclass:testcase01

  initial begin
    testcase01 testcase01handle;
    testcase01handle=new();

    env=new(srpif);
    env.gen.custom_trans=testcase01handle;
    env.main(count);
  end


endprogram:test
```

<u>Explanation:</u>

The Verilog code defines a test program that leverages a reusable environment module to generate test cases. It creates a specific test case, testcase01, with constraints on its input values. The environment module likely handles the generation of input stimuli based on the test case definition and interacts with the Design Under Test (DUT). The count parameter controls the number of test cases to be generated.

<u>Driver.sv file</u>

```
class driver;
  mailbox gen2driv, driv2sb;
  virtual srp_if.DRIVER srpif;
  transaction d_trans;
  event driven;
```

```
  function new(mailbox gen2driv, driv2sb , virtual srp_if.DRIVER srpif, event driven);
    this.gen2driv=gen2driv;
    this.srpif=srpif;
    this.driven=driven;
    this.driv2sb=driv2sb;
  endfunction


  task main(input int count);
    repeat(count) begin
      d_trans=new();
      gen2driv.get(d_trans);

      @(srpif.driver_cb);
      srpif.driver_cb.Din <= d_trans.Din;
      srpif.driver_cb.Load <= d_trans.Load;
      srpif.driver_cb.A <= d_trans.A;
      srpif.driver_cb.Left <= d_trans.Left;
      driv2sb.put(d_trans);
      -> driven;
    end

  endtask:main

endclass:driver
```

<u>Explanation:</u>

The Verilog code defines a driver class, which acts as an intermediary between the stimulus generator and the Design Under Test (DUT) in a layered testbench. It receives input stimuli from the generator, drives the DUT's inputs, and notifies the response checker about the applied stimuli.

<u>Environment.sv file</u>

```
`include "generator.sv"
`include "driver.sv"
`include "monitor.sv"
`include "scoreboard.sv"

class environment;
  mailbox gen2driv;
  mailbox driv2sb;
  mailbox mon2sb;
```

```
  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;

  event driven;

  virtual srp_if srpif;

  function new(virtual srp_if srpif);
    this.srpif=srpif;
    gen2driv=new();
    driv2sb=new();
    mon2sb=new();

    gen=new(gen2driv);
    drv=new(gen2driv,driv2sb,srpif.DRIVER,driven);
    mon=new(mon2sb,srpif.MONITOR,driven);
    scb=new(driv2sb,mon2sb);

  endfunction

  task main(input int count);
    fork  gen.main(count);
          drv.main(count);
          mon.main(count);
          scb.main(count);
    join
    $finish;
  endtask:main

endclass:environment
```

Explanation:

The Verilog code defines an environment class that serves as the core of a layered testbench. It instantiates and coordinates the interaction between four key modules: the generator, driver, monitor, and scoreboard. The generator produces test cases, the driver applies them to the DUT, the monitor samples the DUT's outputs, and the scoreboard compares expected and actual results.

Generator.sv file

```
`include "transaction.sv"

class generator;
  mailbox gen2driv;
  transaction g_trans, custom_trans;

  function new(mailbox gen2driv);
    this.gen2driv=gen2driv;
  endfunction

  task main(input int count);
    repeat(count) begin
      g_trans=new();
      g_trans=new custom_trans;
      assert(g_trans.randomize());
      gen2driv.put(g_trans);
    end
  endtask:main

endclass:generator
```

Explanation:

The Verilog code defines a generator class responsible for generating test cases within a layered testbench. It creates transactions, randomizes their values, and sends them to the next stage (likely the driver) for further processing. This class plays a crucial role in providing a consistent stream of test cases to exercise the Design Under Test (DUT) and ensure its proper functionality.

Interface.sv file

```
interface srp_if(input clk);
  logic Din,Load,Left;
  logic [15:0] A;
  logic Dout;

  clocking driver_cb @(negedge clk);
    default input #1 output #1;
    output Din,Left,Load,A;
  endclocking

  clocking mon_cb @(negedge clk);
    default input #1 output #1;
```

```
    input Din,Left,Load,A;
    input Dout;
  endclocking

  modport DRIVER (clocking driver_cb, input clk);
  modport MONITOR (clocking mon_cb, input clk);

endinterface
```

Explanation:

The Verilog code defines an interface named srp_if that specifies the communication protocol between the testbench and the Design Under Test (DUT). It defines input and output signals, clocking blocks for driving and monitoring, and modports for exposing the interface to the testbench and DUT.

Monitor.sv file

```
class monitor;
  mailbox mon2sb;
  virtual srp_if.MONITOR srpif;
  transaction m_trans;
  event driven;

  function new(mailbox mon2sb, virtual srp_if.MONITOR srpif, event driven);
    this.mon2sb=mon2sb;
    this.srpif=srpif;
    this.driven=driven;
  endfunction

  task main(input int count);
    @(driven);
    @(srpif.mon_cb);
    repeat(count) begin
      m_trans=new();
      @(posedge srpif.clk);
      m_trans.Dout=srpif.mon_cb.Dout;
      mon2sb.put(m_trans);
    end
  endtask:main


endclass:monitor
```

Explanation:

The Verilog code defines a monitor class, which is part of a layered testbench. This class is responsible for monitoring the outputs of the Design Under Test (DUT) and sending the captured data to the scoreboard. It receives the DUT's output values through a virtual interface and packages them into transaction objects, which are then sent to the scoreboard for comparison with expected values.

Scoreboard.sv file

```
class scoreboard;
  mailbox driv2sb;
  mailbox mon2sb;
  bit [15:0] t;   //t=temporary register
  logic [2:0] c;  //combination=c
  real mc [8] ='{default: 64'b0};  //mc=maximum coverage
  real f [8] ='{default: 64'b0};   //f=fail
  real p [8] ='{default: 64'b0};   //p=pass
  real pmc [8], pf [8], pp [8];

  integer i;

  transaction d_trans;
  transaction m_trans;

  event driven;

  function new(mailbox driv2sb, mon2sb);
    this.driv2sb=driv2sb;
    this.mon2sb=mon2sb;
  endfunction

  task main(input int count);
    $display("------------------Scoreboard Test Starts--------------------");
    repeat(count) begin
      m_trans=new();
      mon2sb.get(m_trans);
      report();

      if((!d_trans.Load) && (m_trans.Dout != d_trans.Dout)  )
        begin
        $display("Failed : Din=%d Left=%d A=%b Load=%d Expected out=%d
Resulted out=%d",d_trans.Din,d_trans.Left,d_trans.A,d_trans.Load,d_trans.Dout,
m_trans.Dout);
          c={d_trans.Load,d_trans.Din,d_trans.Left};
    case (c)
```

```
     //begin
      3'b000: f[0]=f[0]+1;
      3'b001: f[1]=f[1]+1;
      3'b010: f[2]=f[2]+1;
      3'b011: f[3]=f[3]+1;
      3'b100: f[4]=f[4]+1;
      3'b101: f[5]=f[5]+1;
      3'b110: f[6]=f[6]+1;
      3'b111: f[7]=f[7]+1;
    // end
    endcase
      end
      else
        begin
        $display("passed : Din=%d Left=%d A=%b Load=%d Expected out=%d
Resulted out=%d",d_trans.Din,d_trans.Left,d_trans.A,d_trans.Load,d_trans.Dout,
m_trans.Dout);
          c={d_trans.Load,d_trans.Din,d_trans.Left};
    case (c)
     //begin
      3'b000: p[0]=p[0]+1;
      3'b001: p[1]=p[1]+1;
      3'b010: p[2]=p[2]+1;
      3'b011: p[3]=p[3]+1;
      3'b100: p[4]=p[4]+1;
      3'b101: p[5]=p[5]+1;
      3'b110: p[6]=p[6]+1;
      3'b111: p[7]=p[7]+1;
    // end
    endcase
    end
    end

    for (i = 0; i<8 ; i++)
      begin
        pmc[i]= (mc[i]*100)/count;
        pp [i]= (p[i]*100)/mc[i];
        pf [i]= (f[i]*100)/mc[i];

      end

    for(i = 0; i <8 ; i++)
      begin
        $display ("Type- (Load,Din,Left)->'%3b'=%0.0f cases with percentage=%f. Pass
rate= %f, Fail rate= %f ",i, mc[i],pmc[i],pp[i],pf[i] );
      end


    $display("------------------Scoreboard Test Ends--------------------");
  endtask:main
```

```
  task report();
   d_trans=new();
   driv2sb.get(d_trans);
   //maximum coverage find out
   c={d_trans.Load,d_trans.Din,d_trans.Left};
   case (c)
    //begin
     3'b000: mc[0]=mc[0]+1;
     3'b001: mc[1]=mc[1]+1;
     3'b010: mc[2]=mc[2]+1;
     3'b011: mc[3]=mc[3]+1;
     3'b100: mc[4]=mc[4]+1;
     3'b101: mc[5]=mc[5]+1;
     3'b110: mc[6]=mc[6]+1;
     3'b111: mc[7]=mc[7]+1;
   // end
   endcase


   if (d_trans.Load)
        begin
          t=d_trans.A; // load
        end
   else if (d_trans.Left)
       begin
          // Left shift
        d_trans.Dout = t[15];
        t = {t[14:0], d_trans.Din};
       end
       else
        begin
          // Right shift
          d_trans.Dout = t[0];
          t = {d_trans.Din, t[15:1]};
         end
  endtask:report
endclass:scoreboard
```

Explanation:

The scoreboard class is a crucial component of the layered testbench. It receives expected and actual outputs from the driver and monitor modules, respectively. It then compares these values, calculates coverage metrics, and generates a detailed report, including pass/fail rates and coverage information.

Transaction.sv file

```
class transaction;
  rand bit  Din;
  rand bit Left;
  rand bit Load;
  rand bit [15:0] A;
  bit  Dout;

endclass:transaction
```

Explanation:

The transaction class defines a data structure used to represent a single test case within the layered testbench. It contains fields for the input signals (Din, Left, Load, and A) as well as the expected output signal (Dout). The rand keyword indicates that these fields should be randomly assigned values when creating a new transaction object.

Maximum coverage

Count=100

Output:

```
Type- (Load,Din,Left)->'000'=9 cases with percentage=9.000000. Pass rate= 100.00
0000, Fail rate= 0.000000
Type- (Load,Din,Left)->'001'=10 cases with percentage=10.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'010'=15 cases with percentage=15.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'011'=11 cases with percentage=11.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'100'=10 cases with percentage=10.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'101'=17 cases with percentage=17.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'110'=14 cases with percentage=14.000000. Pass rate= 100.
000000, Fail rate= 0.000000
Type- (Load,Din,Left)->'111'=14 cases with percentage=14.000000. Pass rate= 100.
000000, Fail rate= 0.000000
------------------Scoreboard Test Ends--------------------
Simulation complete via $finish(1) at time 1015 NS + 1
../testbench/environment.sv:39     $finish;
ncsim> exit
[vlsi15@CadenceServer3 run]$
```

Basically we have tested the output using different count values. We foundbetter coverage at count = 100. Here each type shoud have have around 12 cases. And from the output, we can see that too.

Output of layered testbench:

```
passed :  Din=1 Left=1 A=0001000010101101 Load=0 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0101000010110111 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=1100011001110111 Load=1 Expected out=0  Resulted out=0
passed :  Din=1 Left=0 A=1101111010001000 Load=0 Expected out=1  Resulted out=1
passed :  Din=1 Left=1 A=0010011010110110 Load=1 Expected out=0  Resulted out=1
passed :  Din=1 Left=0 A=1010001000010100 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1010001110010110 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0001010111100111 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=1 A=0111110101100101 Load=0 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=1110001111110100 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=1 A=0100011110000101 Load=0 Expected out=0  Resulted out=0
passed :  Din=0 Left=0 A=0101010000111111 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1010010110000000 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0011000001010111 Load=1 Expected out=0  Resulted out=0
passed :  Din=1 Left=0 A=1111010010111011 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0010111010000000 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0111101010000111 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=0 A=1001000110110010 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=0 A=1010001110101010 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0110110111101001 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=0 A=0011000110000101 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=0 A=1111001011100010 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0001101001011001 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=1 A=0101001000001011 Load=0 Expected out=1  Resulted out=1
passed :  Din=1 Left=1 A=1101111000100100 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=0 A=1111001100001101 Load=0 Expected out=1  Resulted out=1
passed :  Din=1 Left=0 A=0011100101100011 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=0100100100011100 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=1 A=0010001111111000 Load=1 Expected out=0  Resulted out=1
passed :  Din=1 Left=0 A=0101101101000011 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1010100101100111 Load=1 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1100101110100101 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=1 A=1000011000000111 Load=0 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=0101011010011000 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=0 A=1111010011000010 Load=1 Expected out=0  Resulted out=0
passed :  Din=0 Left=0 A=0011111100010001 Load=0 Expected out=0  Resulted out=0
passed :  Din=0 Left=1 A=1111101111100010 Load=1 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1011011000011111 Load=0 Expected out=1  Resulted out=1
passed :  Din=0 Left=0 A=1101111101101010 Load=1 Expected out=0  Resulted out=1
passed :  Din=1 Left=0 A=1101111101110011 Load=1 Expected out=0  Resulted out=1
passed :  Din=1 Left=0 A=0010000001001111 Load=1 Expected out=0  Resulted out=1
passed :  Din=0 Left=0 A=0111000101110101 Load=1 Expected out=0  Resulted out=1
passed :  Din=0 Left=0 A=0010010110010100 Load=1 Expected out=0  Resulted out=1
passed :  Din=0 Left=0 A=0001000110101010 Load=0 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=0101000010010111 Load=1 Expected out=0  Resulted out=0
passed :  Din=1 Left=1 A=1111100100111111 Load=1 Expected out=0  Resulted out=0
```

```
passed : Din=1 Left=0 A=0100110010010100 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=1011110010000110 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0011000100110110 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=0 A=0000011011010001 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0001000000101000 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=1101101011100100 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=1 A=0101111011111000 Load=0 Expected out=1  Resulted out=1
passed : Din=0 Left=0 A=1100001110101000 Load=0 Expected out=1  Resulted out=1
passed : Din=1 Left=1 A=1000011100100001 Load=1 Expected out=0  Resulted out=1
passed : Din=1 Left=1 A=1111010111111010 Load=1 Expected out=0  Resulted out=1
passed : Din=1 Left=1 A=1100011010010100 Load=0 Expected out=1  Resulted out=1
passed : Din=0 Left=0 A=1010111011000000 Load=0 Expected out=1  Resulted out=1
passed : Din=1 Left=1 A=0110001101000111 Load=0 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0101101000101000 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0001110011010111 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=1 A=1101011010010100 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=1000110000101000 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0001110000100001 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=1 A=0011101011010101 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0001101010110101 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0111010100110010 Load=0 Expected out=1  Resulted out=1
passed : Din=0 Left=0 A=0000000101111110 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=1100010110111000 Load=0 Expected out=1  Resulted out=1
passed : Din=1 Left=1 A=1011011101001000 Load=1 Expected out=0  Resulted out=1
passed : Din=1 Left=1 A=0001001000001001 Load=0 Expected out=1  Resulted out=1
passed : Din=1 Left=1 A=0111101011011011 Load=1 Expected out=0  Resulted out=1
passed : Din=1 Left=0 A=0100010000101110 Load=0 Expected out=1  Resulted out=1
passed : Din=0 Left=0 A=1010101111101010 Load=1 Expected out=0  Resulted out=1
passed : Din=1 Left=0 A=1011001111000000 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=1 A=0100100100111111 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=0 A=0000111010111000 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0111110010110010 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=0 A=1001000010001000 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=1001010100100111 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0100111110010010 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=1 A=0110001110110101 Load=0 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0111101101011100 Load=1 Expected out=0  Resulted out=0
passed : Din=1 Left=0 A=0000100010111010 Load=0 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0001111110000101 Load=1 Expected out=0  Resulted out=0
passed : Din=0 Left=1 A=0011101101111101 Load=0 Expected out=0  Resulted out=0
```

So from here, we can say that our designed RTL code is verified as it passed the layered verification.

8 Optimizing Time Constraints:

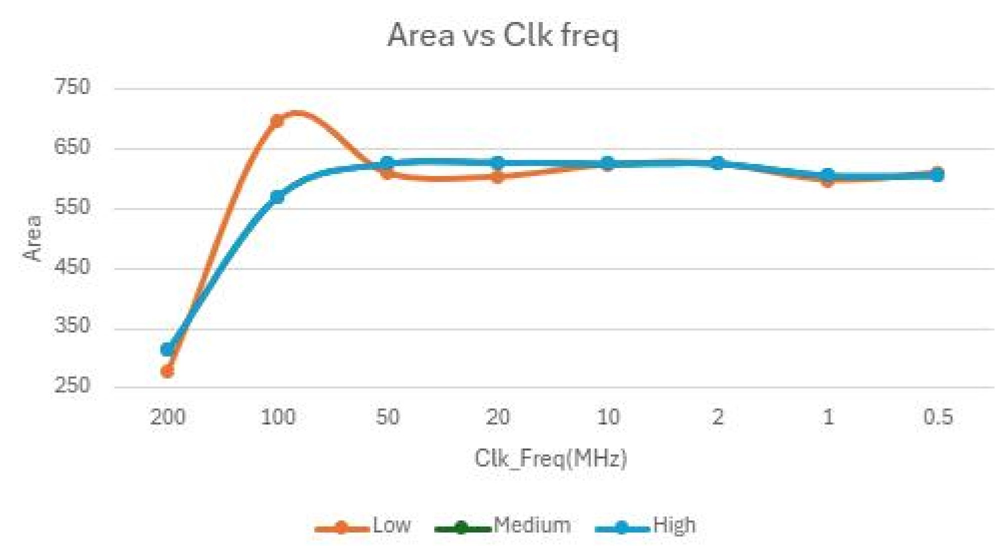
Our code succeeded on layered testbench. Now we need to synthesis the design. But before that, we need to optimize our time contrains in the design. We have to optimize Clock frequency, input and output delay. Other time constraints such as set up time, hold time need not to be optimized as we did not consider reset in our code.

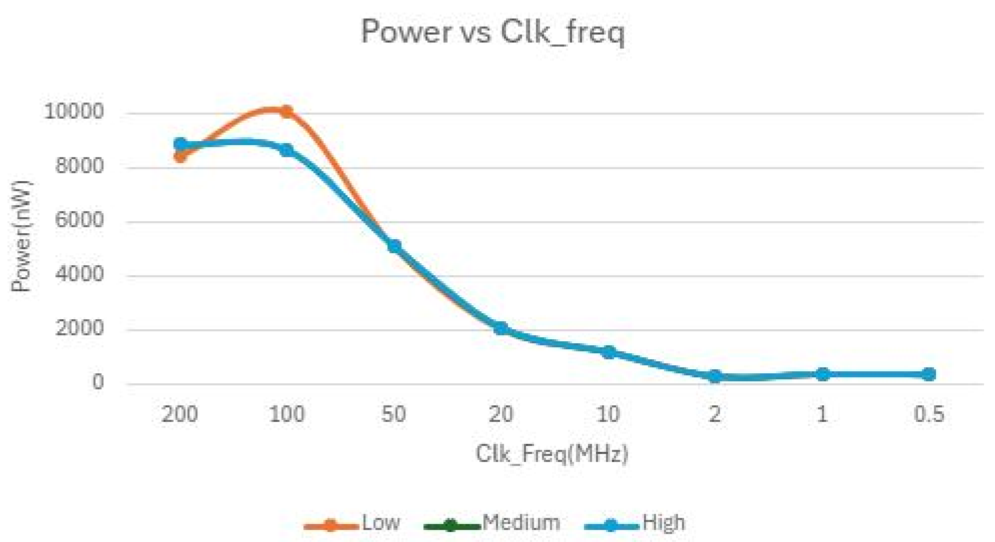
8.1 Optimzing Clock frequency:

We have collected the data of area and power consumed from area and power report for different clock frequencies. We observe that for both high and medium effort of computation, our result is identical.

Clk_Period	Clk_freq(MHz)	Area:	Low	Medium	High	Power:	Low	Medium	High
5	200		277	312	312		8399.836	8821.334	8821.334
10	100		696	569	569		10011.04	8643.366	8643.366
20	50		608	624	624		5051.155	5099.079	5099.079
50	20		603	626	626		2018.974	2068.709	2068.709
100	10		623	624	624		1181.65	1176.025	1176.025
500	2		625	624	624		272.933	272.295	272.295
1000	1		597	604	604		349.747	362.999	362.999
2000	0.5		609	604	604		349.582	341.869	341.869

Now we plot the data for both area and power.

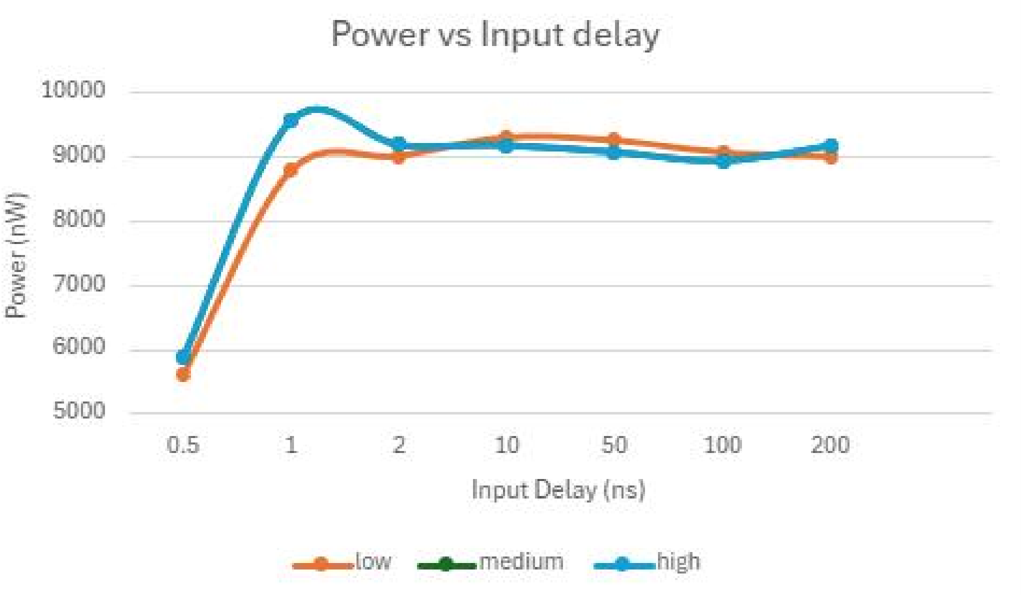




Here, area is only minimum at 200MHz and for all the frequencies, its just averge.
But from the power plot, power is quite high for 200MHz. It is minimum at 2MHz.
And also at 2MHz, area is just average.
Considering all the points, we have taken 2MHz as our optimum clock frequency.

8.2 Optimizing Input Delay

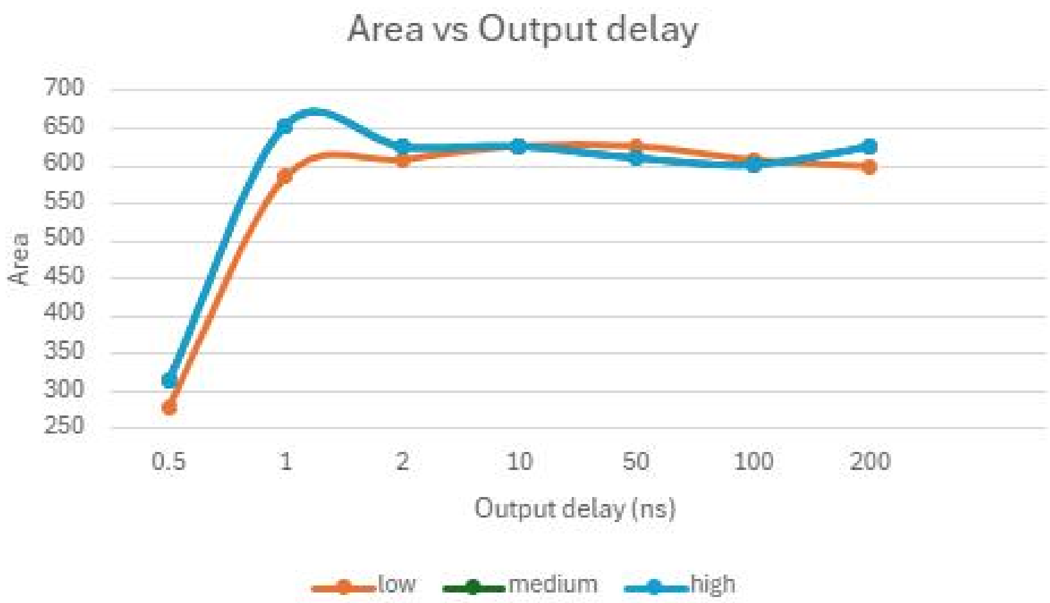
Input Delay	Area	Low	High	Medium	Power	low	medium	high
0.5		277	312	312		5592.294	5864.644	5864.644
1		584	653	653		8777.188	9556.284	9556.284
2		607	624	624		9001.86	9176.802	9176.802
10		625	625	625		9297.295	9173.207	9173.207
50		625	609	609		9248.137	9063.307	9063.307
100		607	600	600		9060.392	8931.56	8931.56
200		598	625	625		8998.764	9167.305	9167.305

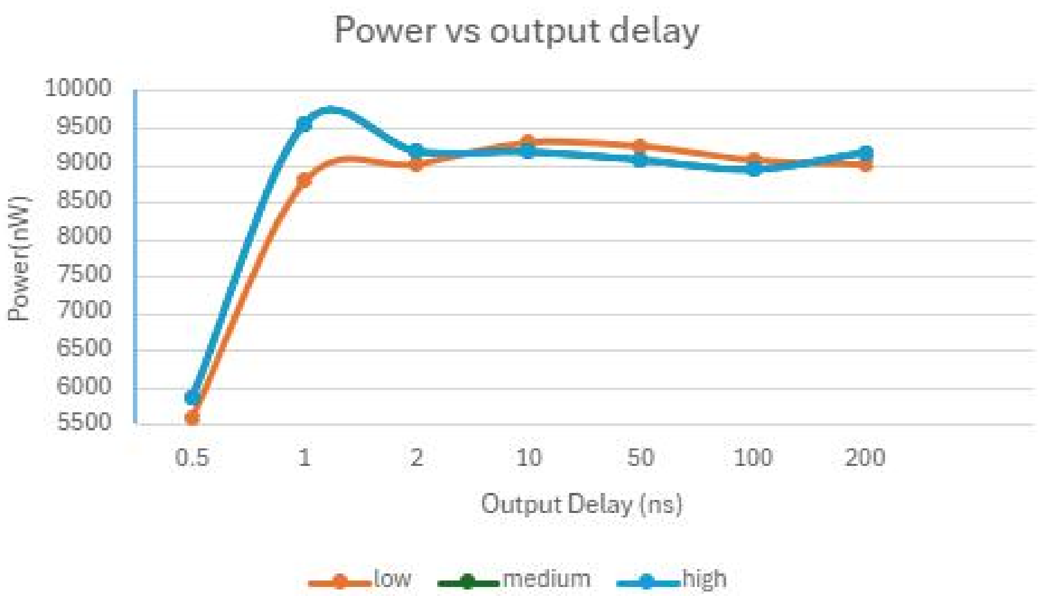


From the plots,area and power is minimum at 0.5 ns. That's why, we have considered 0.5 ns as our optimum input delay.

8.3 Optimizing Output Delay

Output Delay	Area	low	medium	high	power	low	medium	high
0.5		277	312	312		5592.294	5864.644	5864.644
1		584	653	653		8777.188	9556.284	9556.284
2		607	624	624		9001.86	9176.802	9176.802
10		625	625	625		9297.295	9173.207	9173.207
50		625	609	609		9248.137	9063.307	9063.307
100		607	600	600		9060.392	8931.56	8931.56
200		598	625	625		8998.764	9167.305	9167.305





From the plots,area and power is minimum at 0.5 ns. That's why, we have considered 0.5 ns as our optimum output delay

9 Synthesis

A netlist is a description of the connectivity of an electronic circuit which provides nothing more than instances, nets, IO ports and perhaps some attributes. In digital circuit design, Register-Transfer Level (RTL) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals. RTL Logic synthesis is a process by which an abstract form of desired circuit behavior, typically at Register Transfer Level (RTL), is turned into a design implementation in terms of logic gates, typically by a computer program called a synthesis tool. Here, we will use Cadence Genus(TM) Synthesis Solution as a synthesis tool.

9.1 Our Behavioural code (srp.v)

srp.sdc | srp.tcl | srp.v

```verilog
// Code your design here
module srp (clk,Load,Din,Left,Dout,A); //srp=shift registrer project
    input  clk;          // Clock
    input  Load;         //load
    input  Din;
    input [15:0] A;      // parallel load
    input  Left;         // (1: left,0:right)
    output reg Dout;     // Serial output
    reg [15:0] shift_reg; // 16-bit shift register


    always @(posedge clk)
      begin
        if (Load)
          begin
            shift_reg <= A; // parallel load
          end
        else if (Left)
        begin
            // Left shift
          Dout <= shift_reg[15];
          shift_reg <= {shift_reg[14:0], Din};

        end
        else
          begin
            // Right shift
            Dout <= shift_reg[0];
            shift_reg <= {Din, shift_reg[15:1]};

        end
    end
endmodule
```

9.2 Converting to synthesized Netlist (srp.tcl)

srp.sdc | srp.tcl | srp.v

```
#run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false / ;
if {[file exists /proc/cpuinfo]} {
sh grep "model name" /proc/cpuinfo
sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"
##############################################################
### Preset global variables and attributes
##############################################################
set DESIGN srp
set SYN_EFF medium
set MAP_EFF medium
set OPT_EFF medium
# Directory of PDK
#set pdk_dir /home/wsadiq/buet_flow/GPDK045
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
#set_attribute init_lib_search_path $pdk_dir/gsclib045/timing
#set_attribute init_hdl_search_path /home/wsadiq/buet_flow/rtl
set_attribute init_lib_search_path $pdk_dir

#set_attribute init_hdl_search_path ../../rtl
##Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
#set_attribute library "\
slow_vdd1v0_basicCells_hvt.lib \
slow_vdd1v0_basicCells.lib \
slow_vdd1v0_basicCells_lvt.lib"
set_attribute library "\
slow_vdd1v0_basicCells.lib"
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]
# If you dont want to use LVT uncomment this line
```

```
#set_dont_use [get_lib_cells *LVT*]

################################################################
### Load Design
################################################################
###source verilog_files.tcl
read_hdl "\
${DESIGN}.v"
elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
time_info Elaboration
check_design -unresolved
################################################################
### Constraints Setup
################################################################
read_sdc srp.sdc

report timing -encounter >> reports/${DESIGN}_pretim.rpt


################################################################
### Synthesizing to generic
################################################################
syn_generic
puts "Runtime & Memory after 'syn_generic'"
time_info GENERIC
report datapath > reports/${DESIGN}_datapath_generic.rpt
generate_reports -outdir reports -tag generic
write_db -to_file ${DESIGN}_generic.db
report timing -encounter >> reports/${DESIGN}_generic.rpt


##This synthesizes your code
synthesize -to_mapped

## This writes all your files
write -mapped > srp_synth.v

## THESE FILES ARE NOT REQUIRED, THE SDC FILE IS A TIMING FILE
write_script > script
```

Here, we are using for medium computational effort. We have loaded our design. Then we set up our constraints and synthesize to generic. Finally we mapped all files to synth.v file.

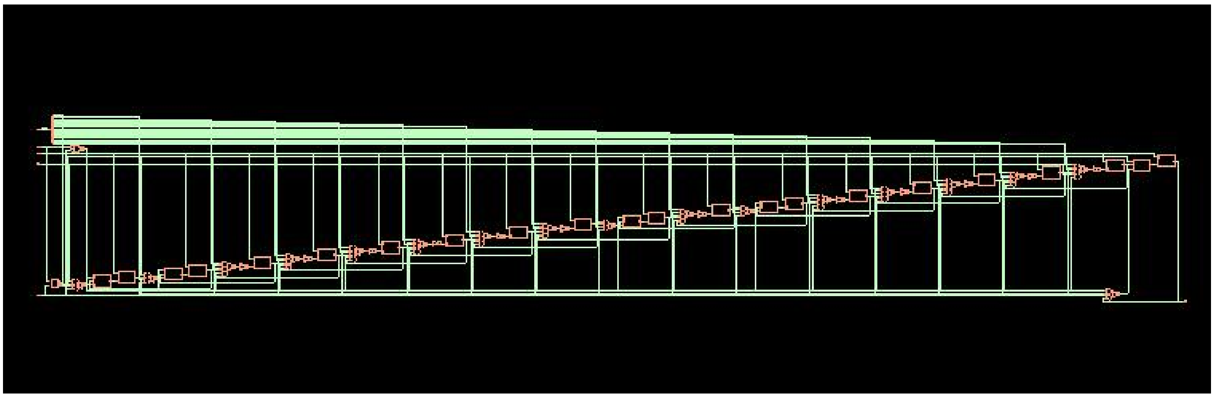
9.3 Synopsys design constraints file (srp.sdc)

srp.sdc srp.tcl srp.v

```
# setting up time units

set_units -time 1ns -capacitance pF

# setting the clock period 10ns, as period = 1/freq, here, freq = 100MHz
set clock_period 500;

set top_module "srp"

set clock_port {clk};

//set reset_port {rst_n};

# setting the input ports in a list to a variable
set input_ports {Din,A,Load,Left} ;

# setting the output ports in a list to a variable
set output_ports {Dout} ;

# define the clocks
create_clock -period ${clock_period} -waveform {0 6} -name func_clk
[get_ports ${clock_port}]

# setting up constraints for the reset signal
set_multicycle_path -setup 1.5 -from [get_ports ${reset_port}]
set_multicycle_path -hold 2 -from [get_ports ${reset_port}]

# Define input delays
set_input_delay 0.5 -clock [get_clocks {func_clk}] ${input_ports}

# Define output delays
set_output_delay 0.5 -clock [get_clocks {func_clk}] ${output_ports}
```

In this code, we have defined all the input output ports. Input and output delay is set to 0.5ns as optimized earlier.Clock period is set to 500ns and so clock frequency is 2MHz.

9.4 Area and Power reports of the optimized design



This is the visual representation of our circuit diagramsynthesized in Cadence. We will be observing the area and power of our design.

<u>Summary Report:</u>

Report Mapped Gates

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)

Generated on: Dec 14 2024 20:48:19

Module: sip

Technology library: slow_vdd1v0 1.0

Operating conditions: PVT_0P9V_125C (balanced_tree)

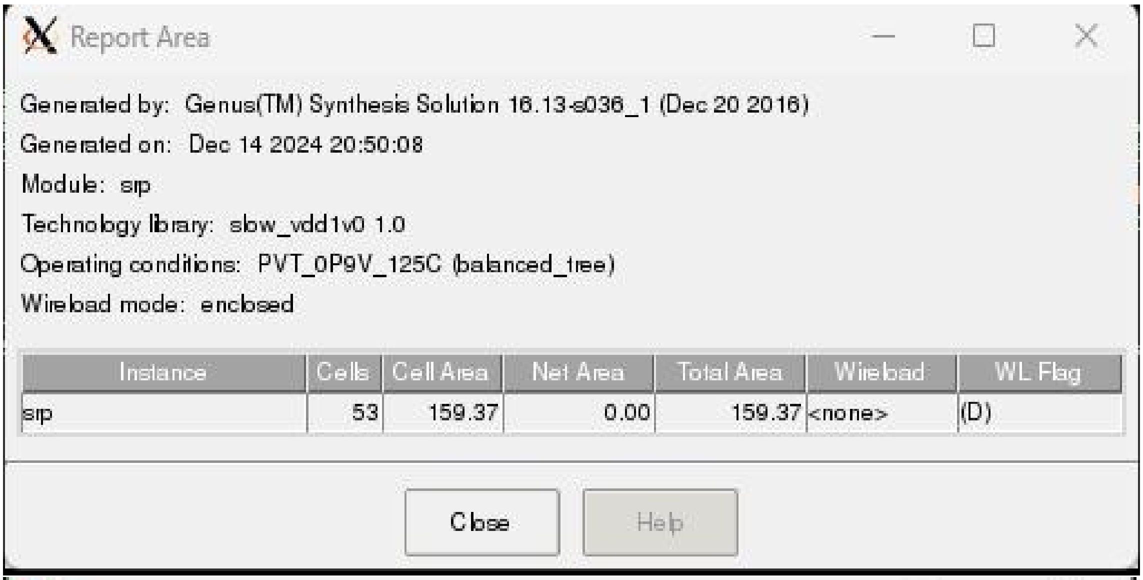
Wireload mode: enclosed

Gate	Instances	Area	Library
AOI222X1	12	36.94	slow_vdd1v0
AOI22XL	5	10.26	slow_vdd1v0
DFFHQX1	17	93.02	slow_vdd1v0
INVXL	12	8.21	slow_vdd1v0
NOR2BX1	1	1.37	slow_vdd1v0
NOR2X1	1	1.03	slow_vdd1v0
OAI2BB1X1	5	8.55	slow_vdd1v0
TOTAL	53	159.38	

Close Help

As we can see here that we need to use 53 gates to implement the circuit. Total area is 159.38 square microns.

Area Report:

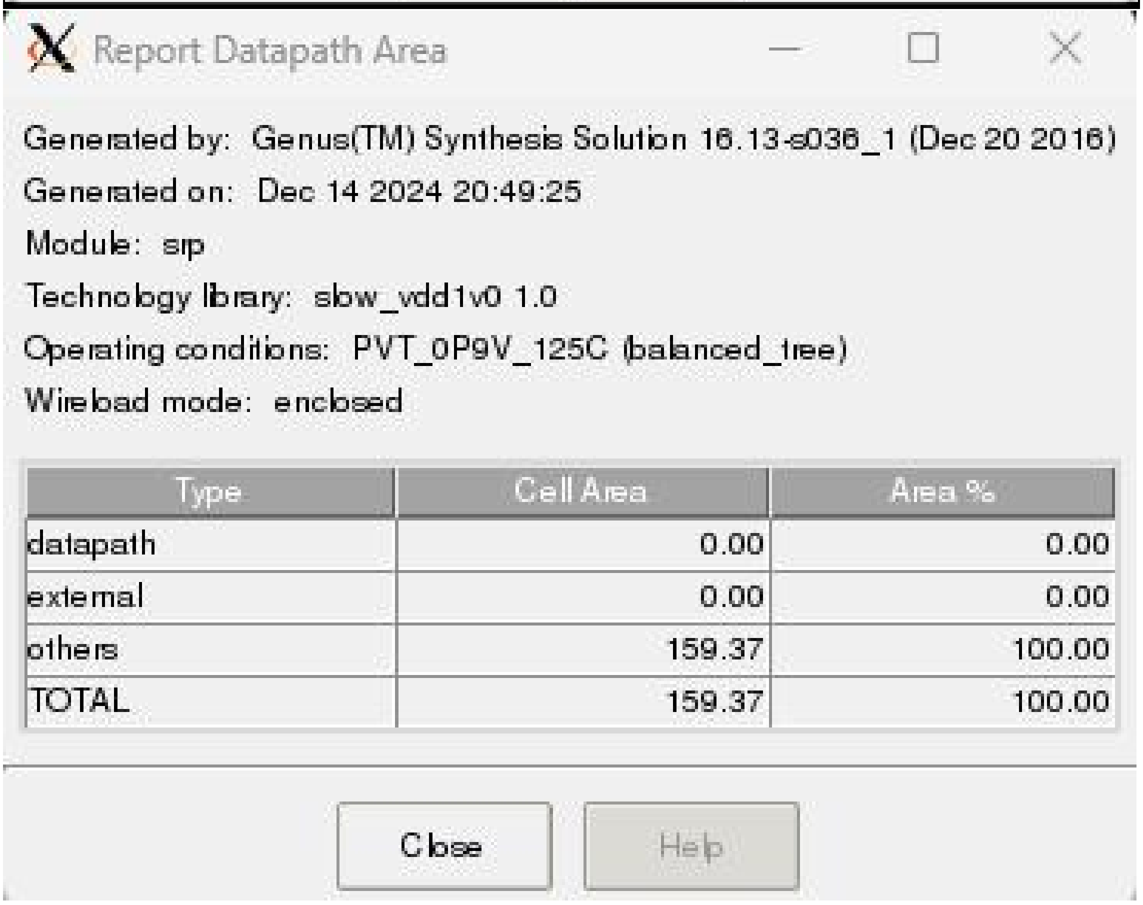


Report Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 14 2024 20:50:08
Module: sip
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
sip	53	159.37	0.00	159.37	<none>	(D)

Close Help



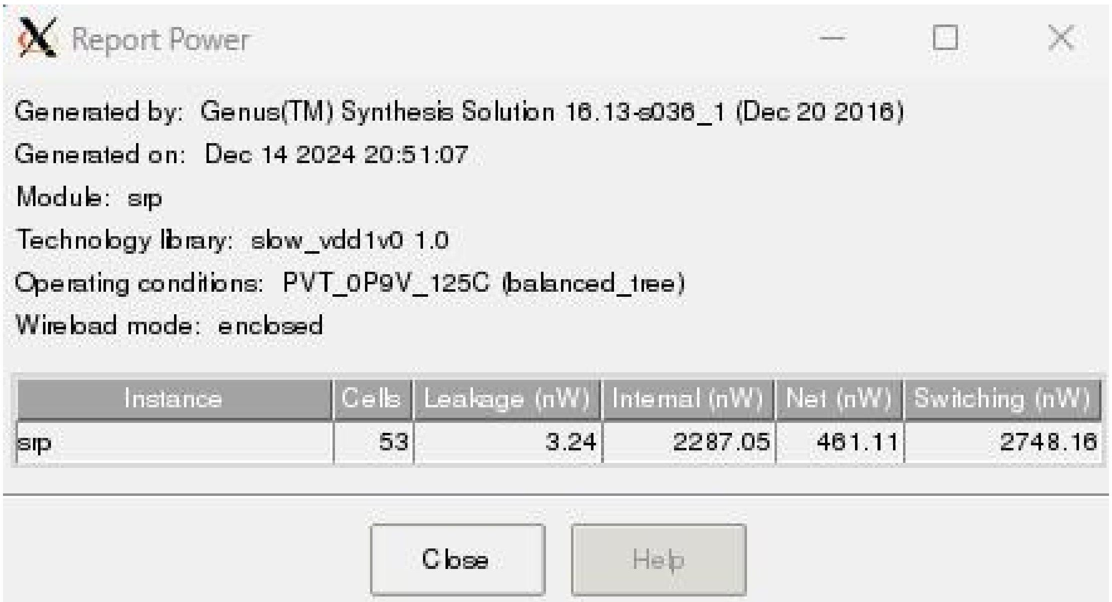
Report Datapath Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 14 2024 20:49:25
Module: sip
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Type	Cell Area	Area %
datapath	0.00	0.00
external	0.00	0.00
others	159.37	100.00
TOTAL	159.37	100.00

Close Help

Detailed Power Report:


Report Power

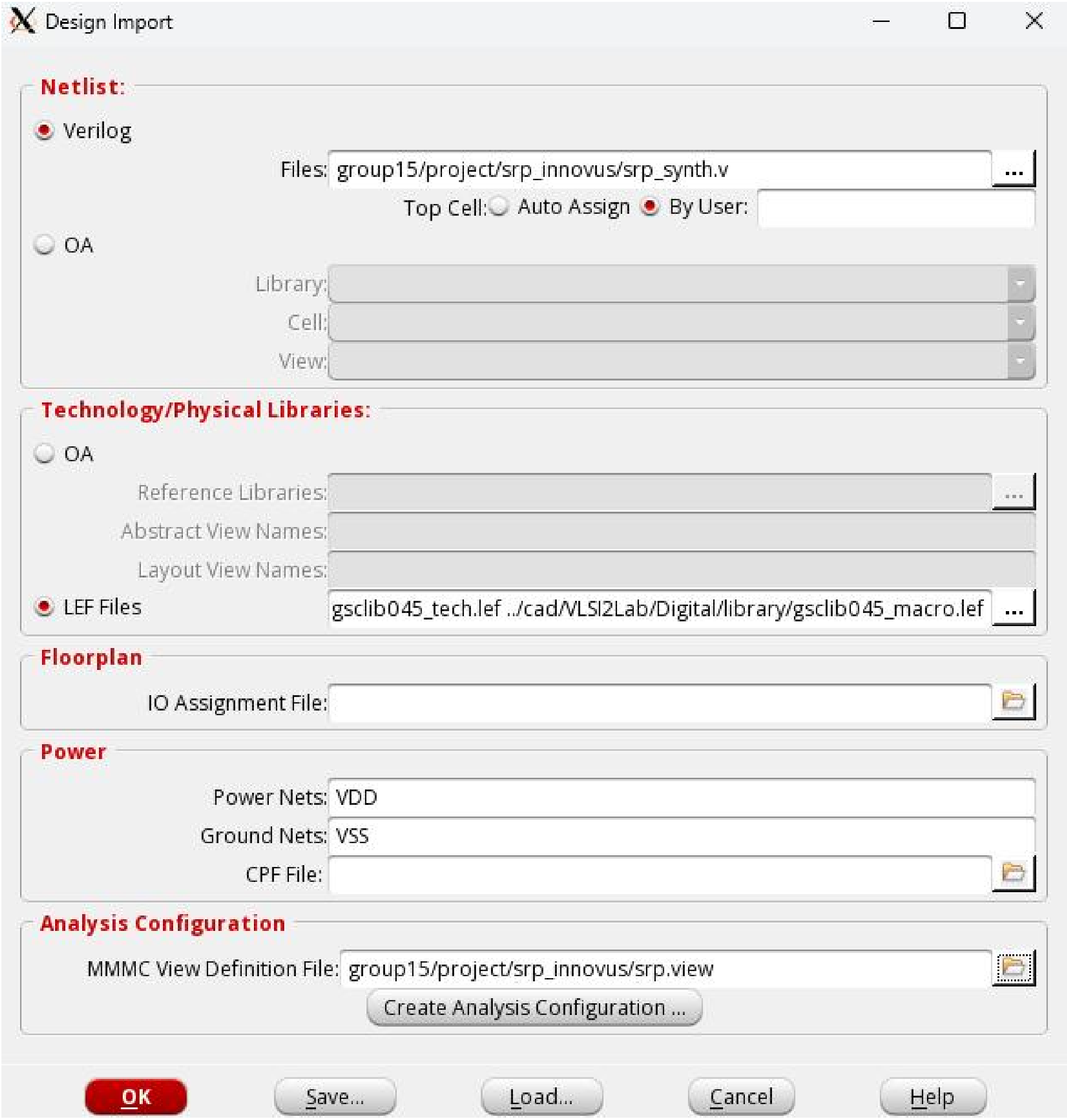
Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 14 2024 20:51:07
Module: sip
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
sip	53	3.24	2287.05	461.11	2748.16

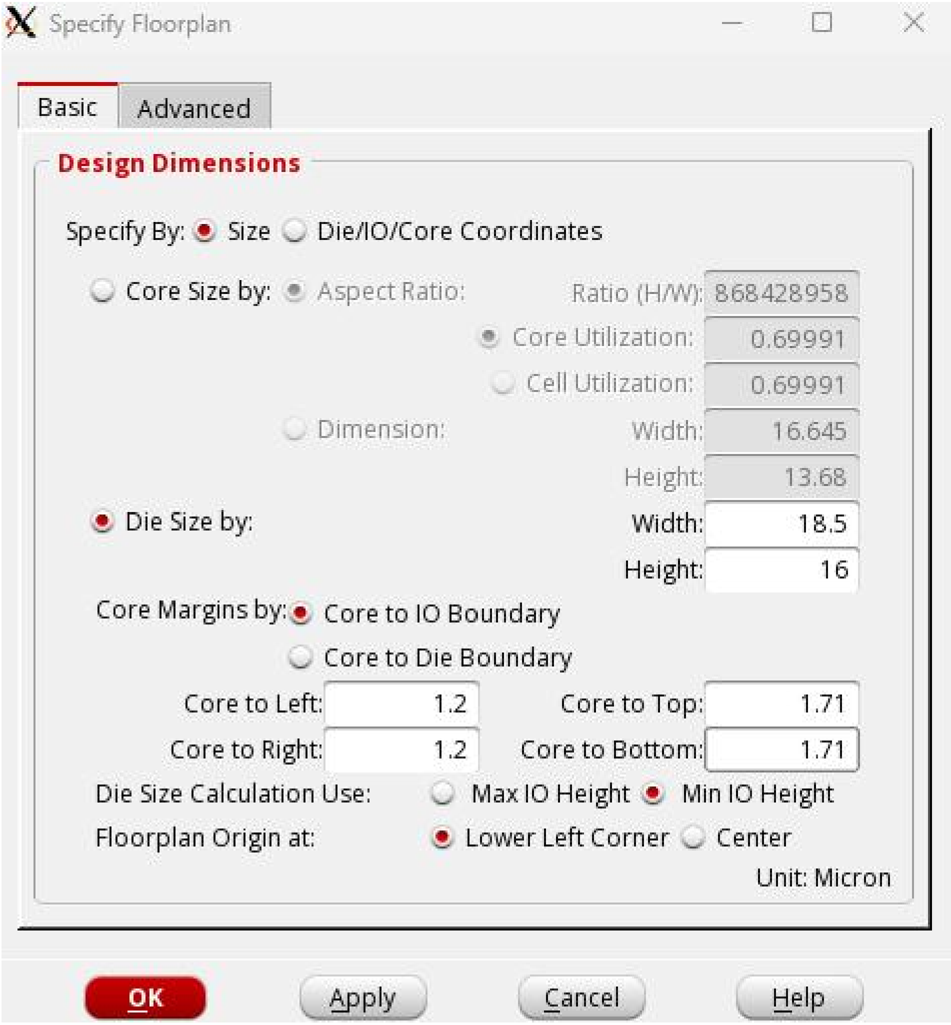
10 Physical Design:

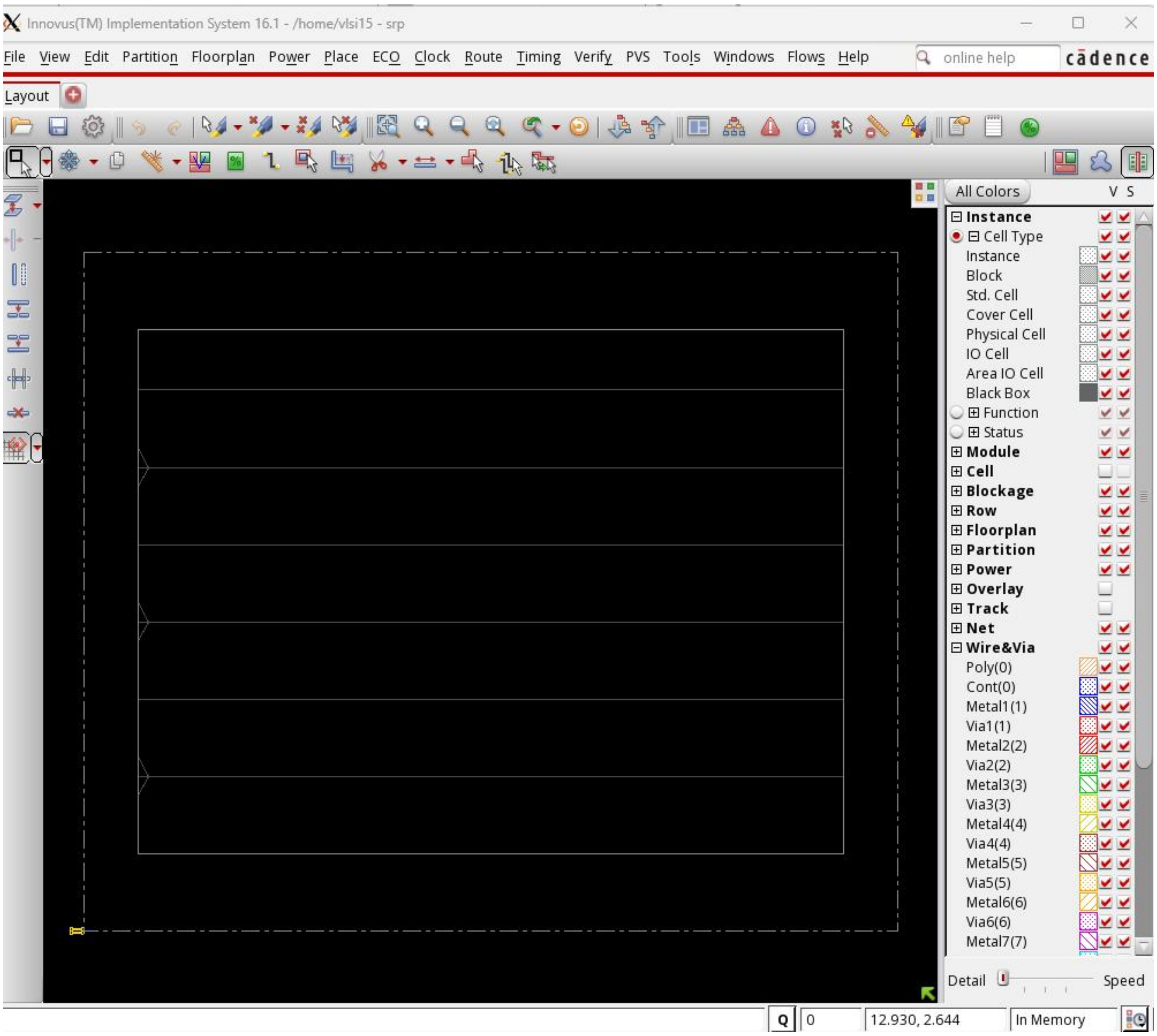
In integrated circuit design, physical design is a step in the standard design cycle which follows after the circuit design. At this step, circuit representations of the components (devices and interconnects) of the design are converted into geometric representations of shapes which, when manufactured in the corresponding layers of materials, will ensure the required functioning of the components. This geometric representation is called integrated circuit layout.
Physical Design is done in Cadence Innovus. At first, we have imported our synthesized and optimized design synth.v file in Innovus. Then we have to follow the following steps to accomplish physical designing.

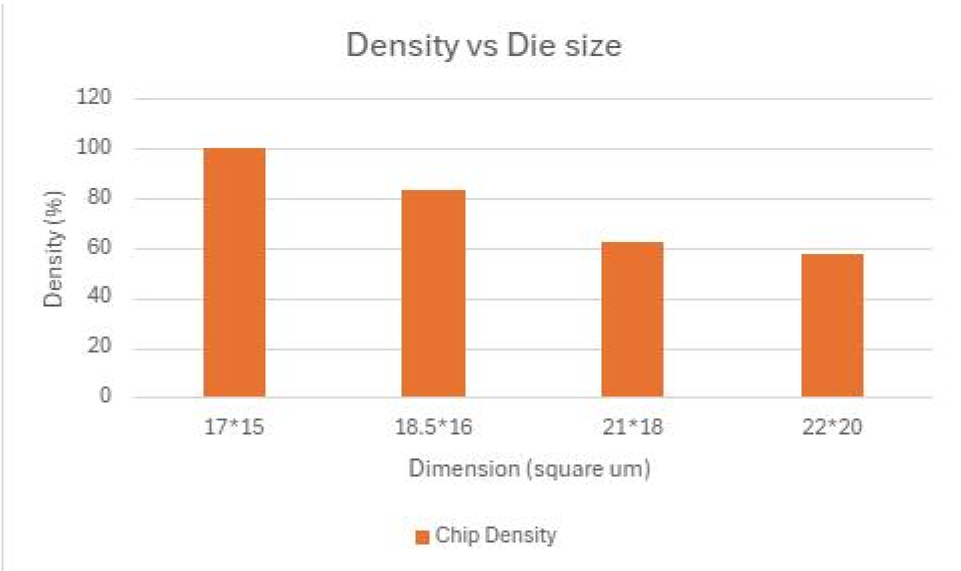
Importing Design:



Floorplaning


Specify Floorplan
Basic
Advanced
Design Dimensions
Specify By: Size Die/IO/Core Coordinates
Core Size by: Aspect Ratio:
Ratio (H/W): 868428958
Core Utilization: 0.69991
Cell Utilization: 0.69991
Dimension:
Width: 16.645
Height: 13.68
Die Size by:
Width: 18.5
Height: 16
Core Margins by: Core to IO Boundary
Core to Die Boundary
Core to Left: 1.2
Core to Top: 1.71
Core to Right: 1.2
Core to Bottom: 1.71
Die Size Calculation Use: Max IO Height Min IO Height
Floorplan Origin at: Lower Left Corner Center
Unit: Micron
OK
Apply
Cancel
Help

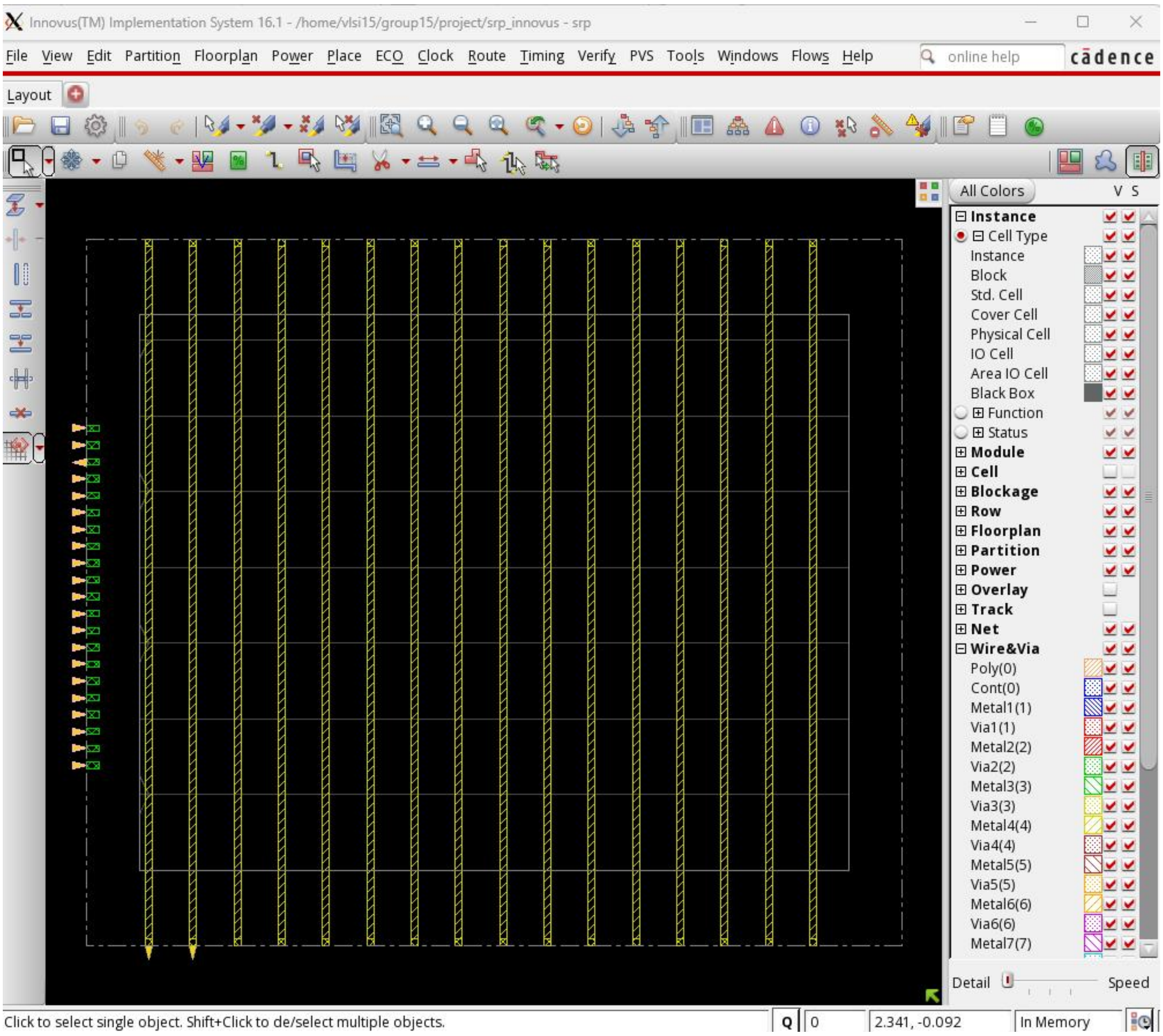

Innovus(TM) Implementation System 16.1 - /home/vlsi15 - srp
File View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS Tools Windows Flows Help
online help
cādence
Layout
All Colors
V S
Instance
Cell Type
Instance
Block
Std. Cell
Cover Cell
Physical Cell
IO Cell
Area IO Cell
Black Box
Function
Status
Module
Cell
Blockage
Row
Floorplan
Partition
Power
Overlay
Track
Net
Wire&Via
Poly(0)
Cont(0)
Metal1(1)
Via1(1)
Metal2(2)
Via2(2)
Metal3(3)
Via3(3)
Metal4(4)
Via4(4)
Metal5(5)
Via5(5)
Metal6(6)
Via6(6)
Metal7(7)
Detail
Speed
Q
0
12.930, 2.644
In Memory


Density vs Die size
120
100
80
60
40
20
0
Density (%)
17*15
18.5*16
21*18
22*20
Dimension (square um)
Chip Density

In order to select the die size, we actually followed trial and error method. Density was found too high for 17*15 and too low for 21*18 and 22*20 square microns. So we chose 18.5*16 square microns as our optimal die size.

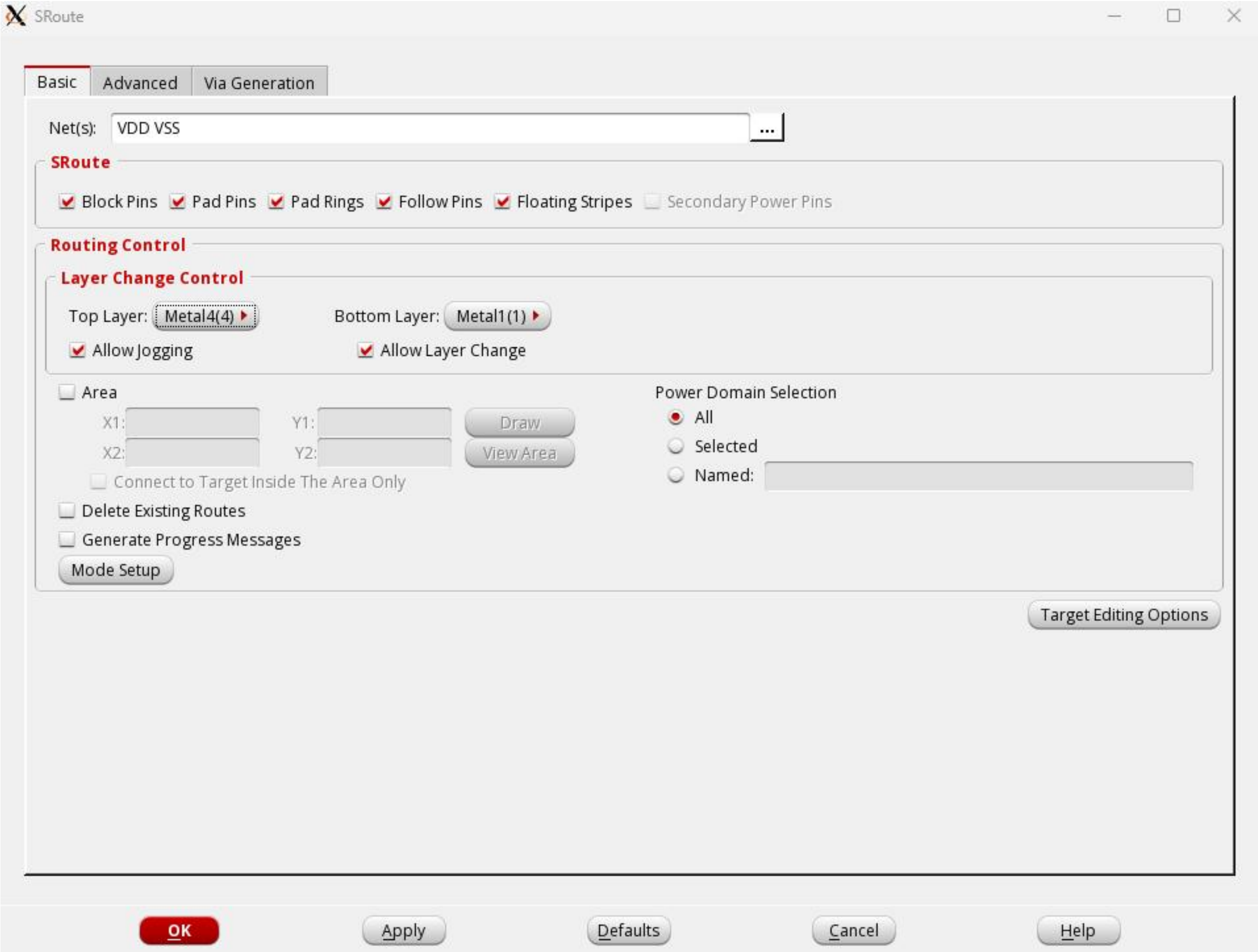
Adding IO ports and stripes

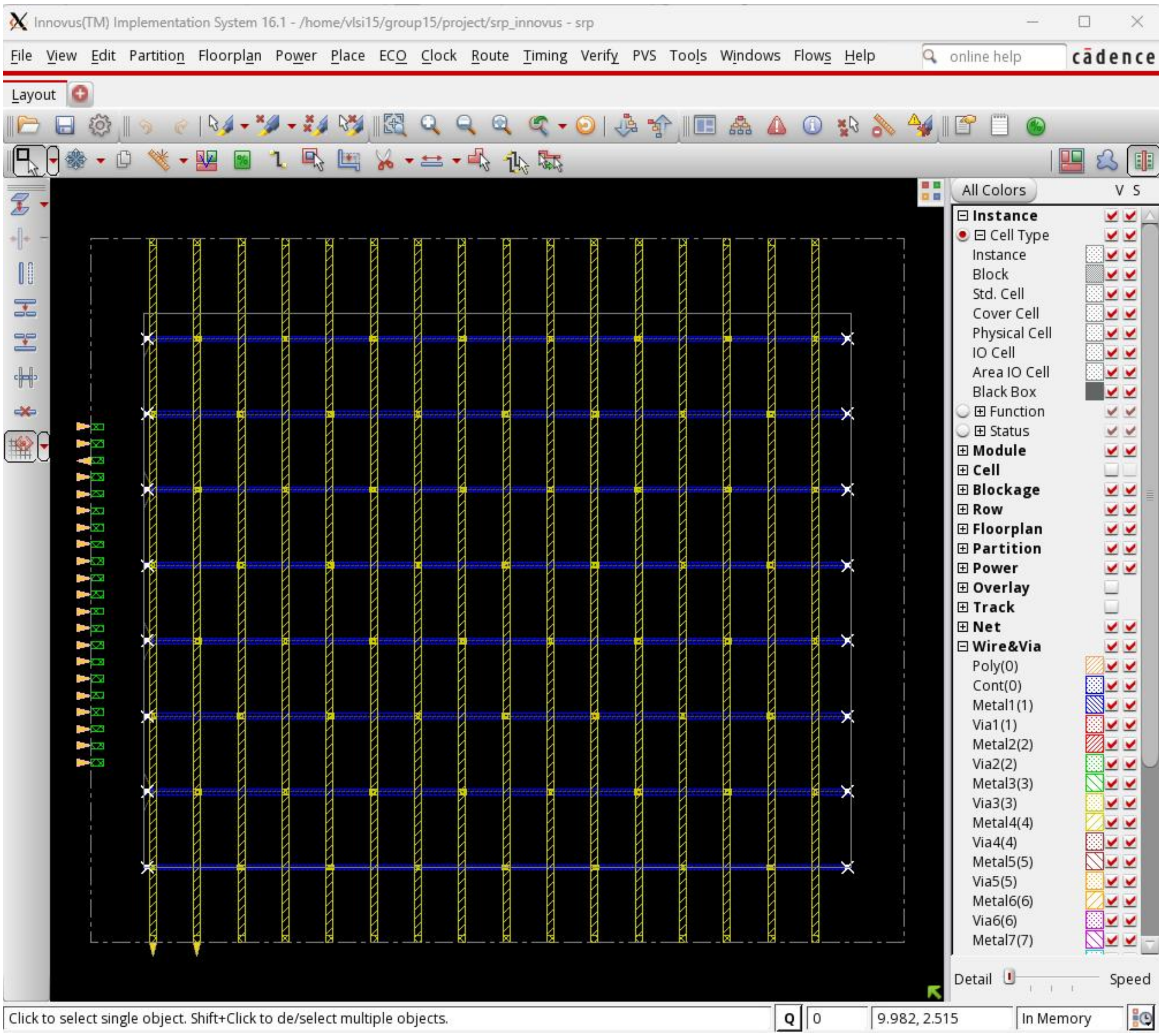
Here, we have tried to take all the pins on the middle of the circuit so that the wires are kept protected by the metal fillings.



Routing and placement:

We added stripes previously using metal(3). For routing we will be using metal(4) and metal(1). After the process, our design looks like this:





Now we need to place standard cells in the design and optimize placement. In this step we want to opmize cell placement,minimize congestion and wire length.

```
-------------------------------------------------------------
     optDesign Final Summary
-------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 0.000   |   N/A   |  0.000  |
|           TNS (ns):| 0.000   |   N/A   |  0.000  |
|    Violating Paths:|   0     |   N/A   |    0    |
|          All Paths:|   0     |   N/A   |    0    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 83.214%
Routing Overflow: 0.00% H and 0.00% V
-------------------------------------------------------------
**optDesign ... cpu = 0:00:06, real = 0:00:06, mem = 1049.7M, totSessionCpu=0:00:39 **
**WARN: (IMPOPT-3195):  Analysis mode has changed.
Type 'man IMPOPT-3195' for more detail.
*** Finished optDesign ***
Removing temporary dont_use automatically set for cells with technology sites with no row.
*** Free Virtual Timing Model ...(mem=1049.7M)
**place_opt_design ... cpu = 0:00:09, real = 0:00:10, mem = 964.0M **
*** Finished GigaPlace ***

*** Summary of all messages that are not suppressed in this session:
Severity  ID               Count  Summary
WARNING   IMPEXT-3530          4  The process node is not set. Use the com...
WARNING   IMPSP-9025           2  No scan chain specified/traced.
WARNING   IMPSP-12502          2  Slack driven placement is disabled becau...
WARNING   IMPOPT-3195          2  Analysis mode has changed.
WARNING   IMPOPT-3564          1  The following cells are set dont_use tem...
*** Message Summary: 11 warning(s), 0 error(s)
```

This is the optimized design summary. We can observe that, the density of the design in 83.214%.

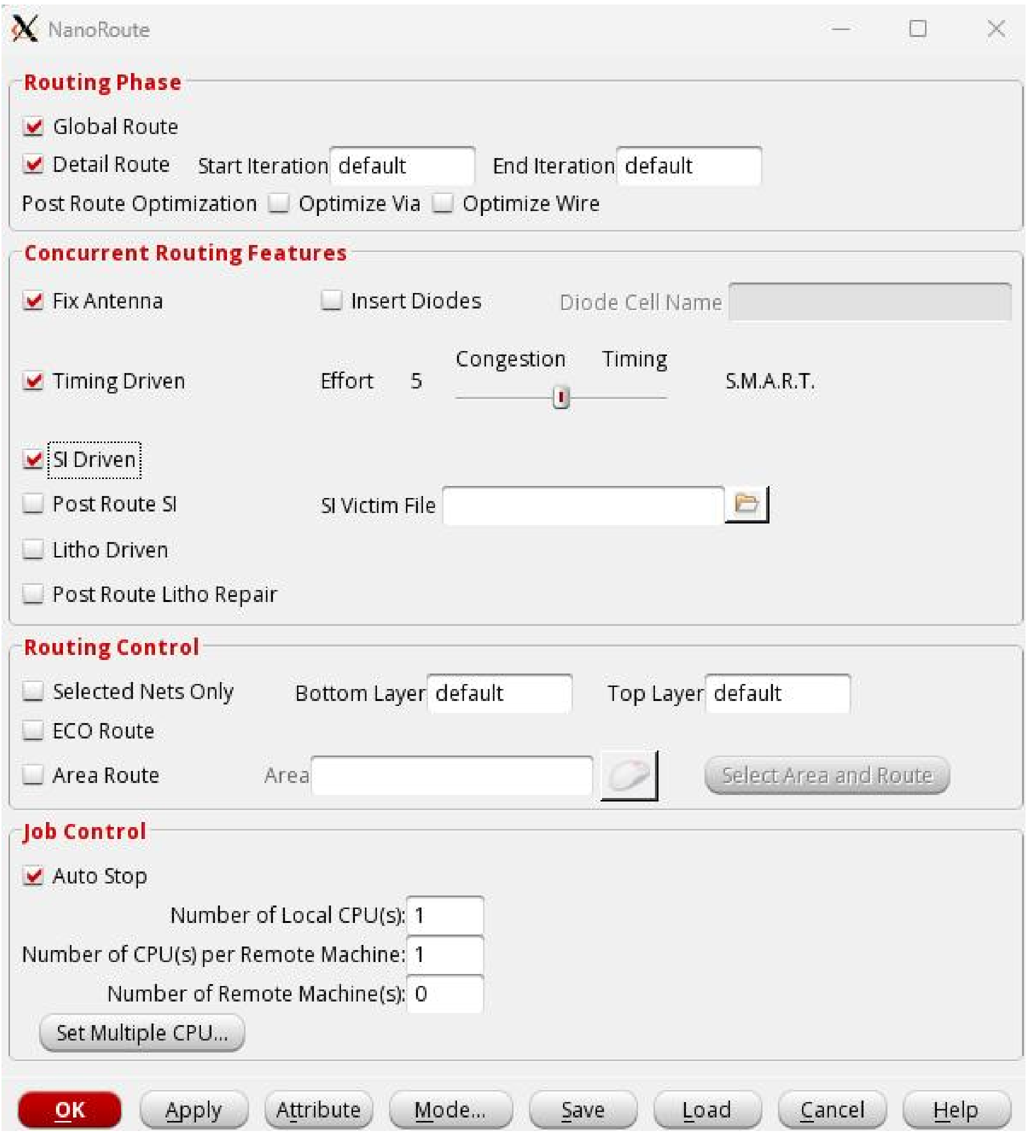
CTS:

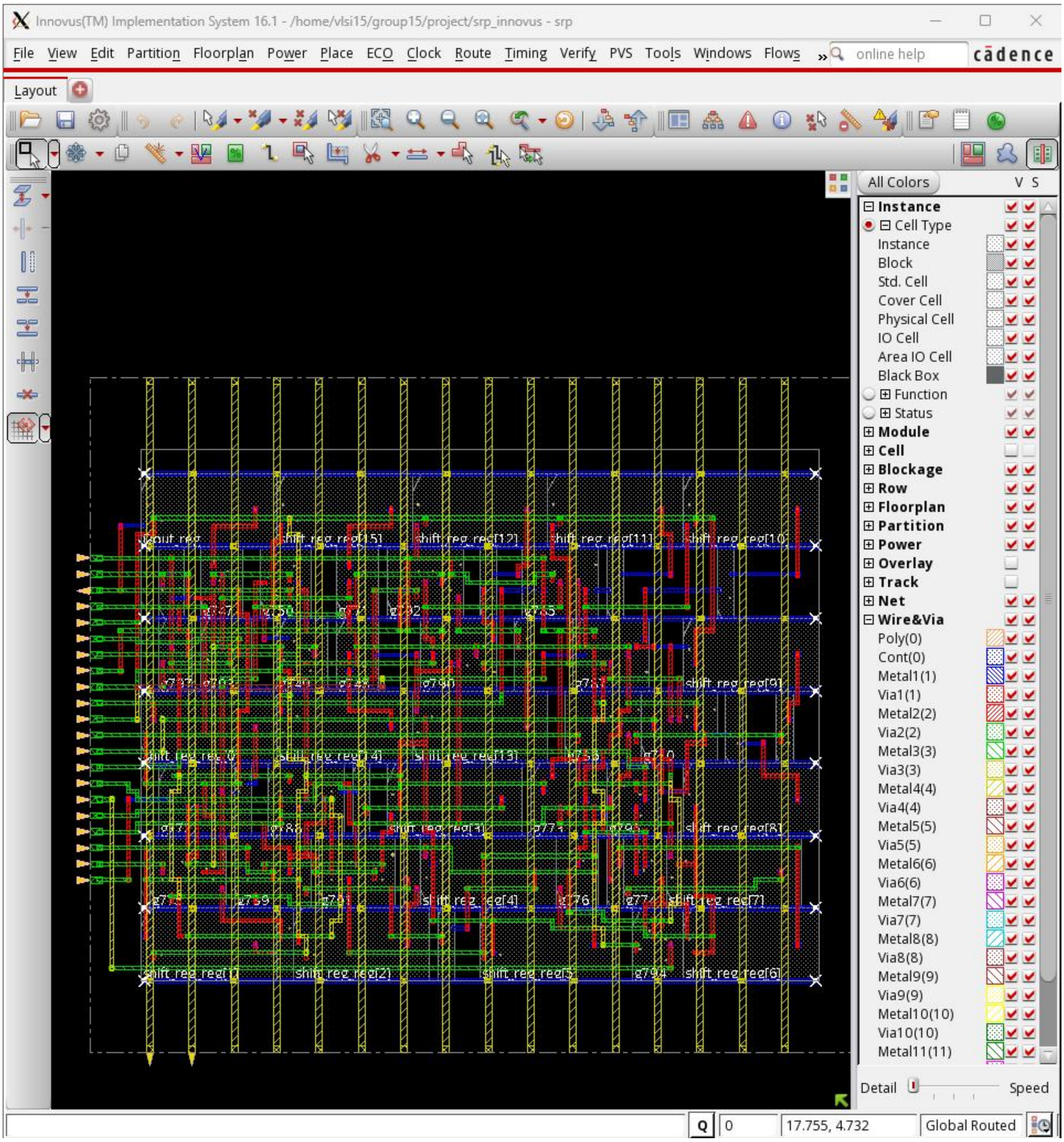
Now we need to synthesize clock tree for this design. After post CTS optimization and timing analysis, we get the following output:

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.000  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|    0    |   N/A   |    0    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 83.214%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 0.17 sec
Total Real time: 1.0 sec
Total Memory Usage: 964.949219 Mbytes
innovus 17> innovus 17>
```

Nano routing

The design is run with global and detail routing using NanoRoute. When running NanoRoute, we enable both the Timing Driven and SI Driven (Signal Integrity) options in the NanoRoute form. These options are important in helping to close timing and preventing crosstalk.


NanoRoute
Routing Phase
Global Route
Detail Route
Start Iteration
default
End Iteration
default
Post Route Optimization
Optimize Via
Optimize Wire
Concurrent Routing Features
Fix Antenna
Insert Diodes
Diode Cell Name
Timing Driven
Effort
5
Congestion
Timing
S.M.A.R.T.
SI Driven
Post Route SI
SI Victim File
Litho Driven
Post Route Litho Repair
Routing Control
Selected Nets Only
Bottom Layer
default
Top Layer
default
ECO Route
Area Route
Area
Select Area and Route
Job Control
Auto Stop
Number of Local CPU(s): 1
Number of CPU(s) per Remote Machine: 1
Number of Remote Machine(s): 0
Set Multiple CPU...
OK
Apply
Attribute
Mode...
Save
Load
Cancel
Help



<u>Post-Route timing and SI Optimization:</u>

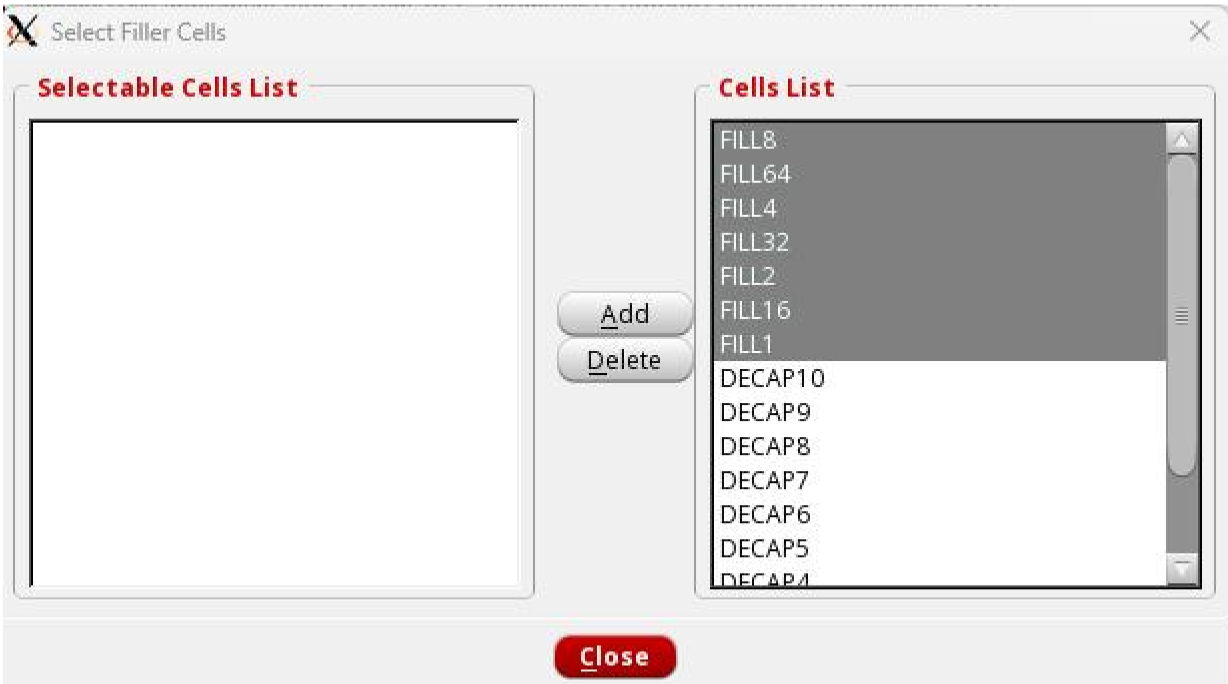
The design is fully routed and timing analysis should be run to analyze timing based on the actual routes.

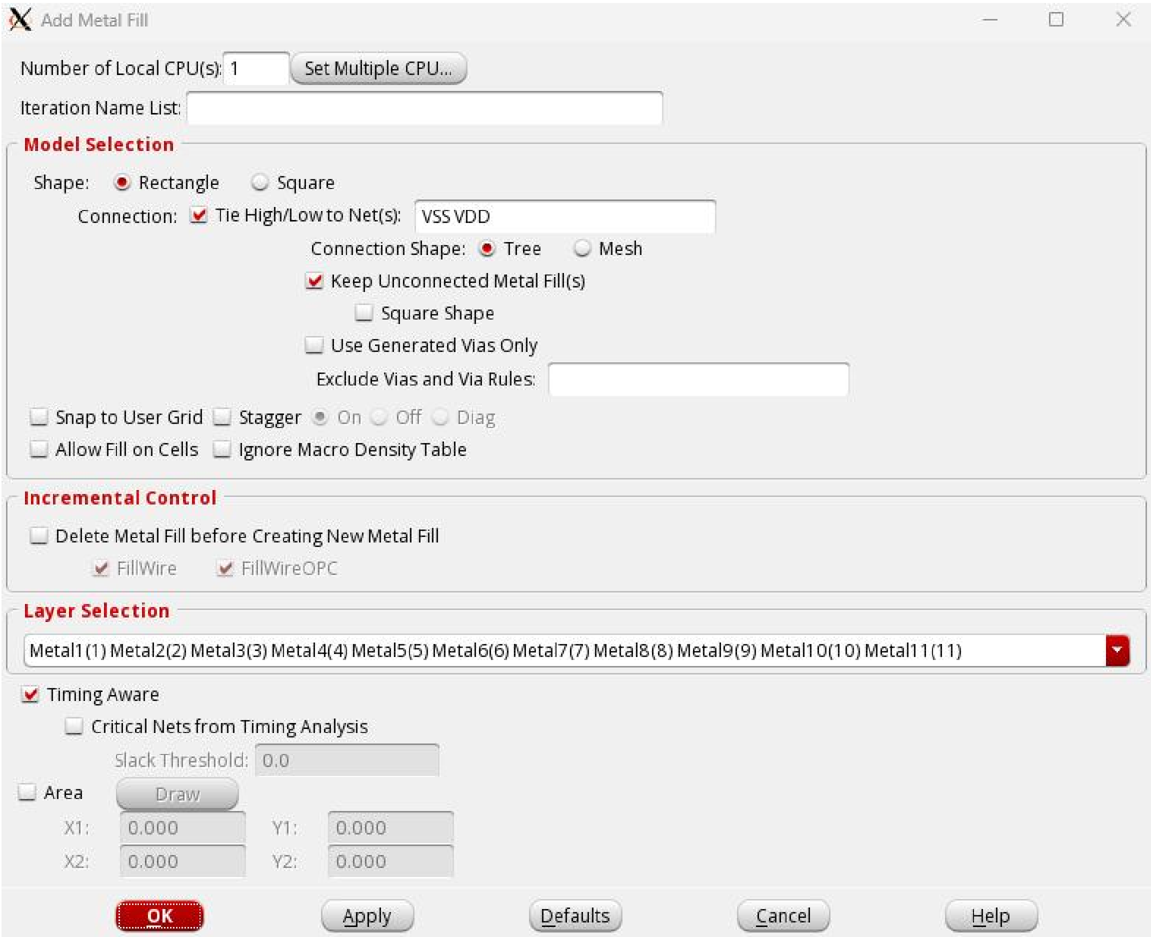
```
          timeDesign Summary
------------------------------------------------------------

Hold  views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.000  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|    0    |   N/A   |    0    |
+--------------------+---------+---------+---------+

Density: 83.214%
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 0.38 sec
Total Real time: 1.0 sec
Total Memory Usage: 1075.464844 Mbytes
Reset AAE Options
innovus 23> innovus 23>
```

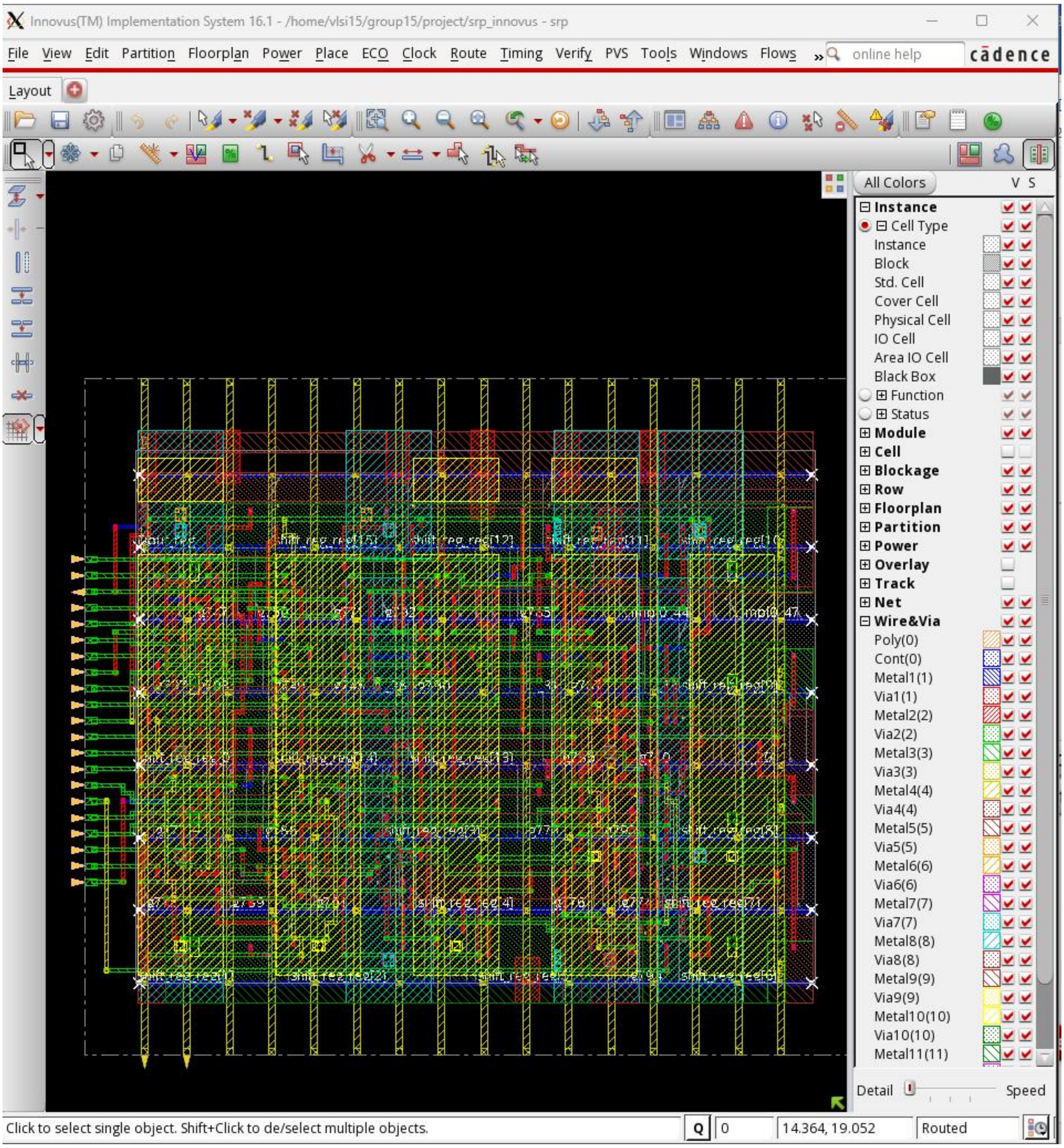
```
     optDesign Final SI Timing Summary
--------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold
Hold  views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.000  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|    0    |   N/A   |    0    |
+--------------------+---------+---------+---------+

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.000  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|          All Paths:|    0    |   N/A   |    0    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 83.214%
Total number of glitch violations: 0
--------------------------------------------------------------
**optDesign ... cpu = 0:00:07, real = 0:00:08, mem = 1135.4M, totSessionCpu=0:01:15 **
 ReSet Options after AAE Based Opt flow
*** Finished optDesign ***
Removing temporary dont_use automatically set for cells with technology sites with no row.
0
innovus 24> innovus 24>
```

Adding filler cell and metal filling:

In order to avoid DRC errors later, it is usually a good idea to place fill cells to fill in the gaps between your placed standard cells. This also provides mechanical stability and substrate uniformity of your chip.


Select Filler Cells
Selectable Cells List
Cells List
FILL8
FILL64
FILL4
FILL32
FILL2
FILL16
FILL1
DECAP10
DECAP9
DECAP8
DECAP7
DECAP6
DECAP5
Add
Delete
Close


Add Metal Fill
Number of Local CPU(s): 1
Set Multiple CPU...
Iteration Name List:
Model Selection
Shape: Rectangle Square
Connection: Tie High/Low to Net(s): VSS VDD
Connection Shape: Tree Mesh
Keep Unconnected Metal Fill(s)
Square Shape
Use Generated Vias Only
Exclude Vias and Via Rules:
Snap to User Grid Stagger On Off Diag
Allow Fill on Cells Ignore Macro Density Table
Incremental Control
Delete Metal Fill before Creating New Metal Fill
FillWire FillWireOPC
Layer Selection
Metal1(1) Metal2(2) Metal3(3) Metal4(4) Metal5(5) Metal6(6) Metal7(7) Metal8(8) Metal9(9) Metal10(10) Metal11(11)
Timing Aware
Critical Nets from Timing Analysis
Slack Threshold: 0.0
Area
Draw
X1: 0.000 Y1: 0.000
X2: 0.000 Y2: 0.000
OK
Apply
Defaults
Cancel
Help



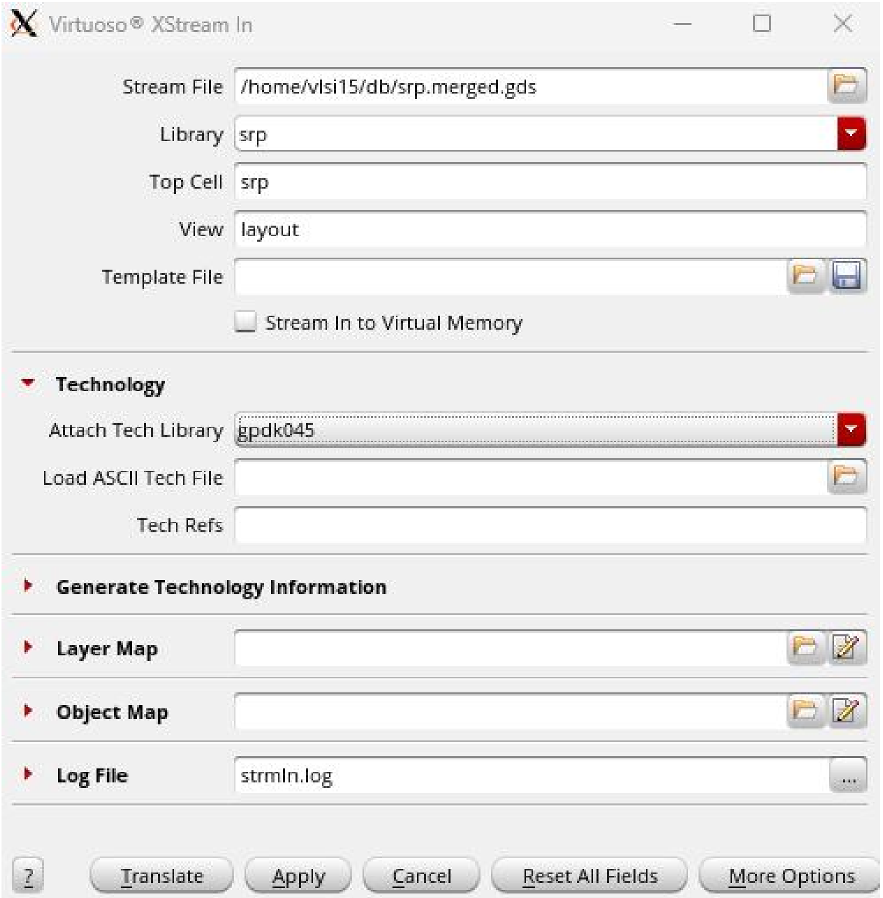
Exporting data:

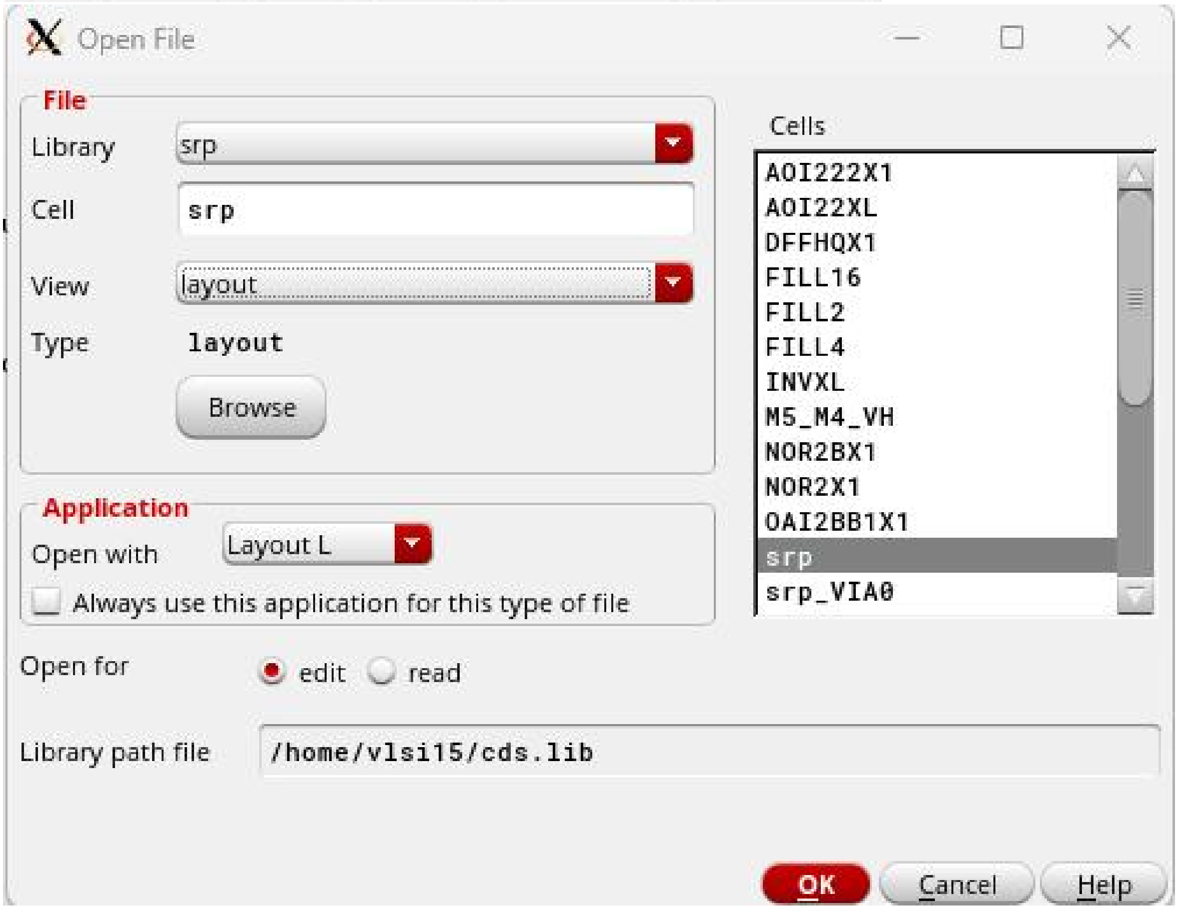
Now we are going to export GDS & netlist to run DRC and LVS in Virtuoso Assura.

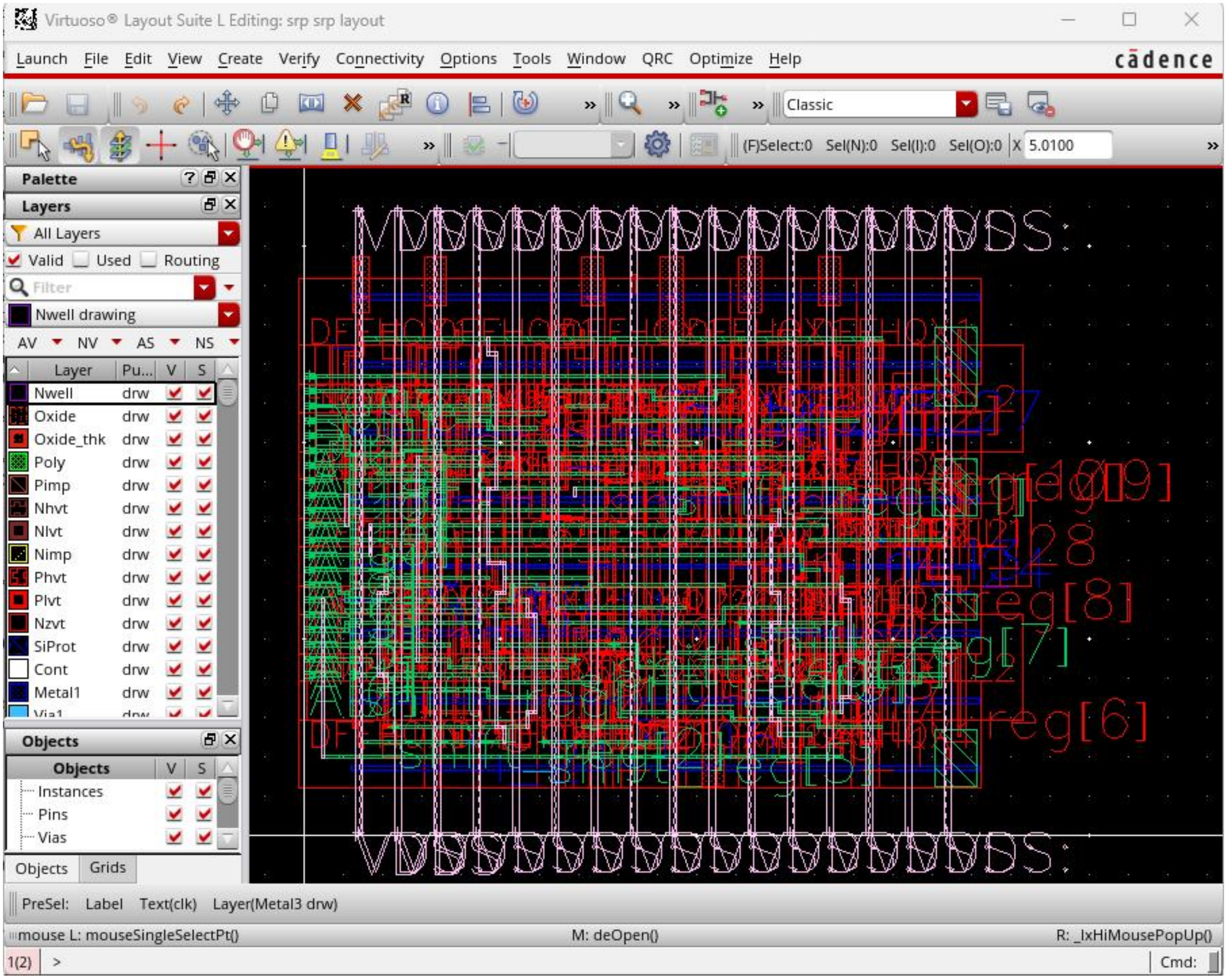
this will generate two files srp.lvs_netlist.vg & srp.merged.gds

DRC results:

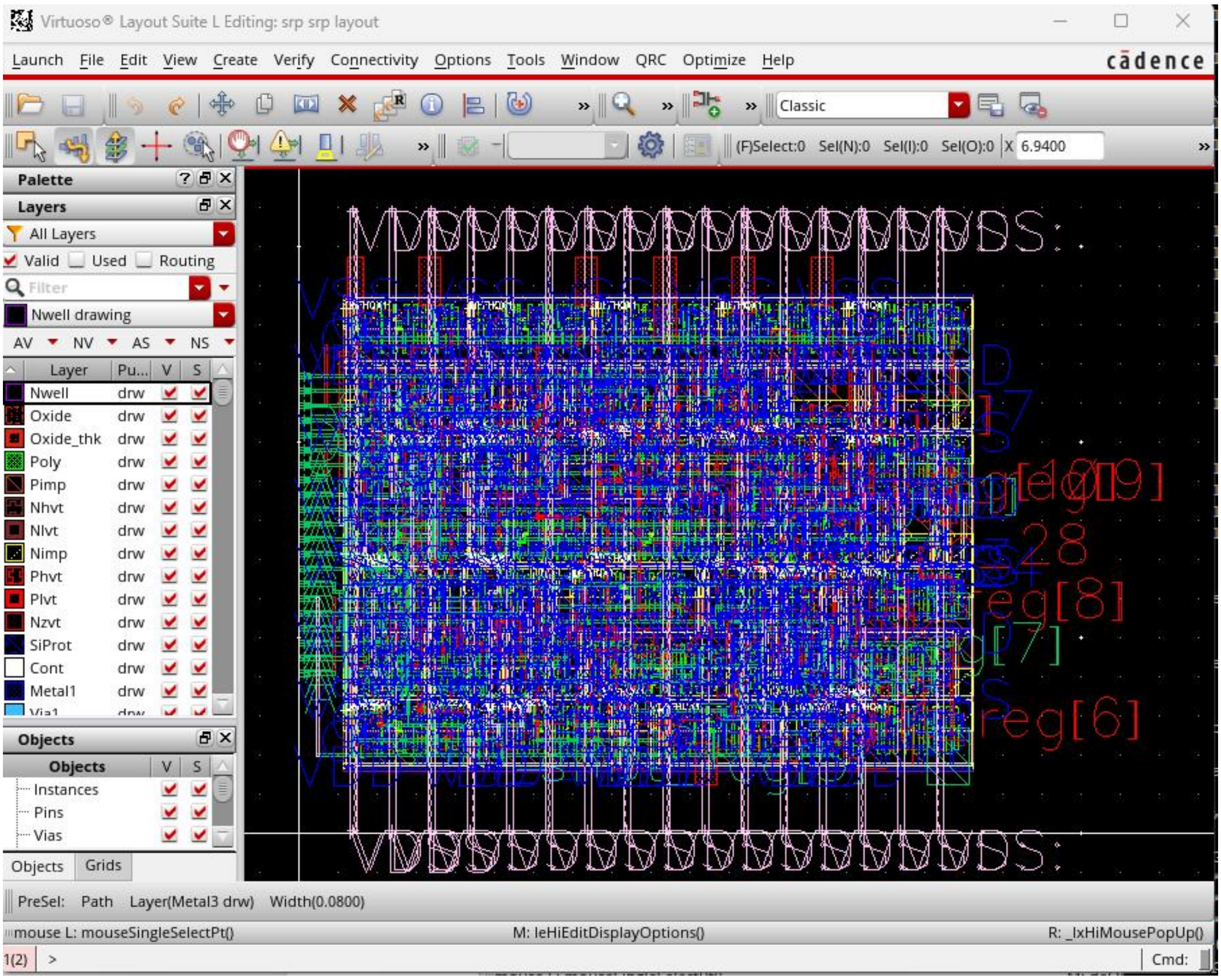
Now we will import this gds srp.merged.gds in Cadence virtuoso and cross check DRC result.

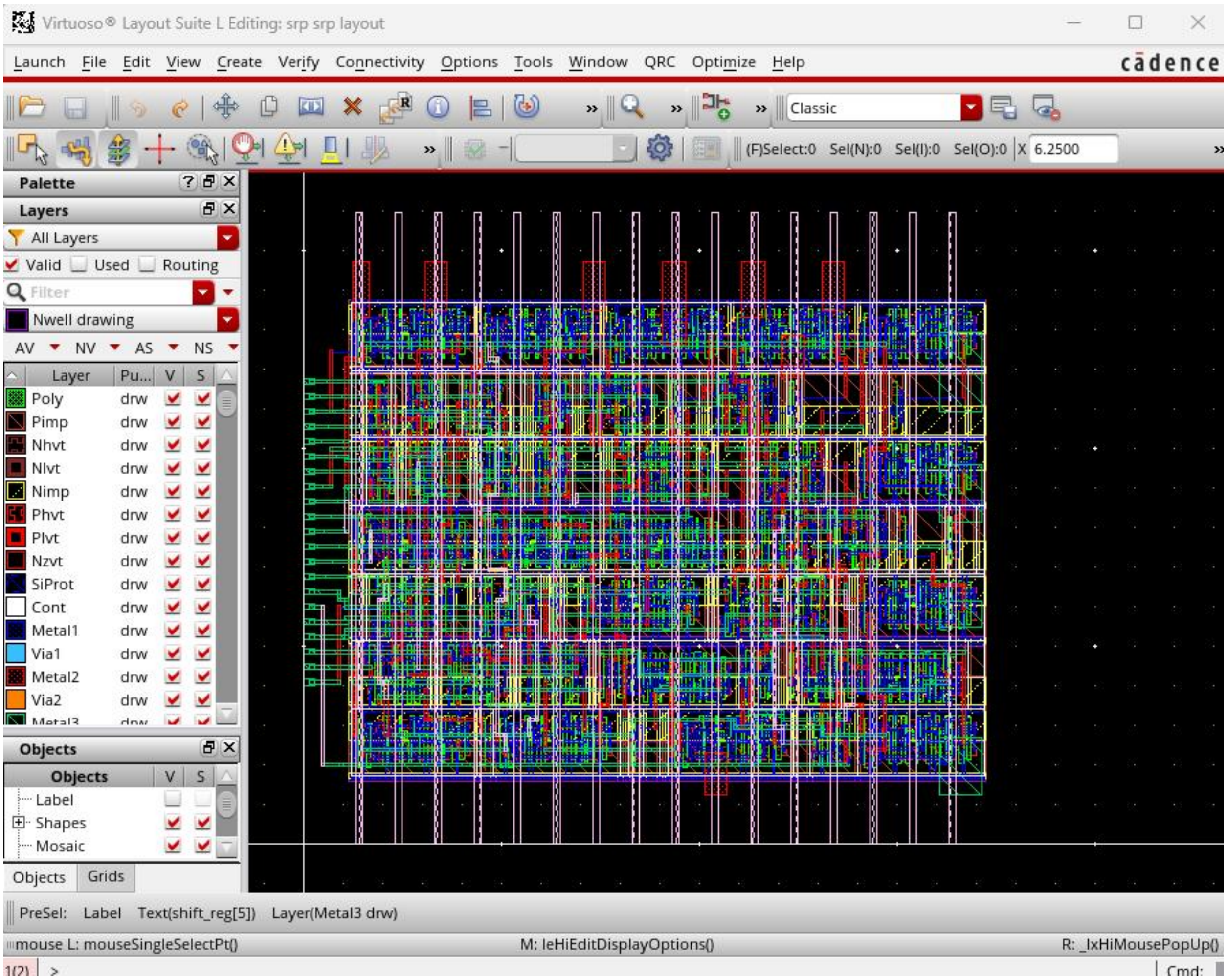

Virtuoso® XStream In
Stream File
/home/vlsi15/db/srp.merged.gds
Library
srp
Top Cell
srp
View
layout
Template File
Stream In to Virtual Memory
Technology
Attach Tech Library
gpdk045
Load ASCII Tech File
Tech Refs
Generate Technology Information
Layer Map
Object Map
Log File
strmIn.log
?
Translate
Apply
Cancel
Reset All Fields
More Options

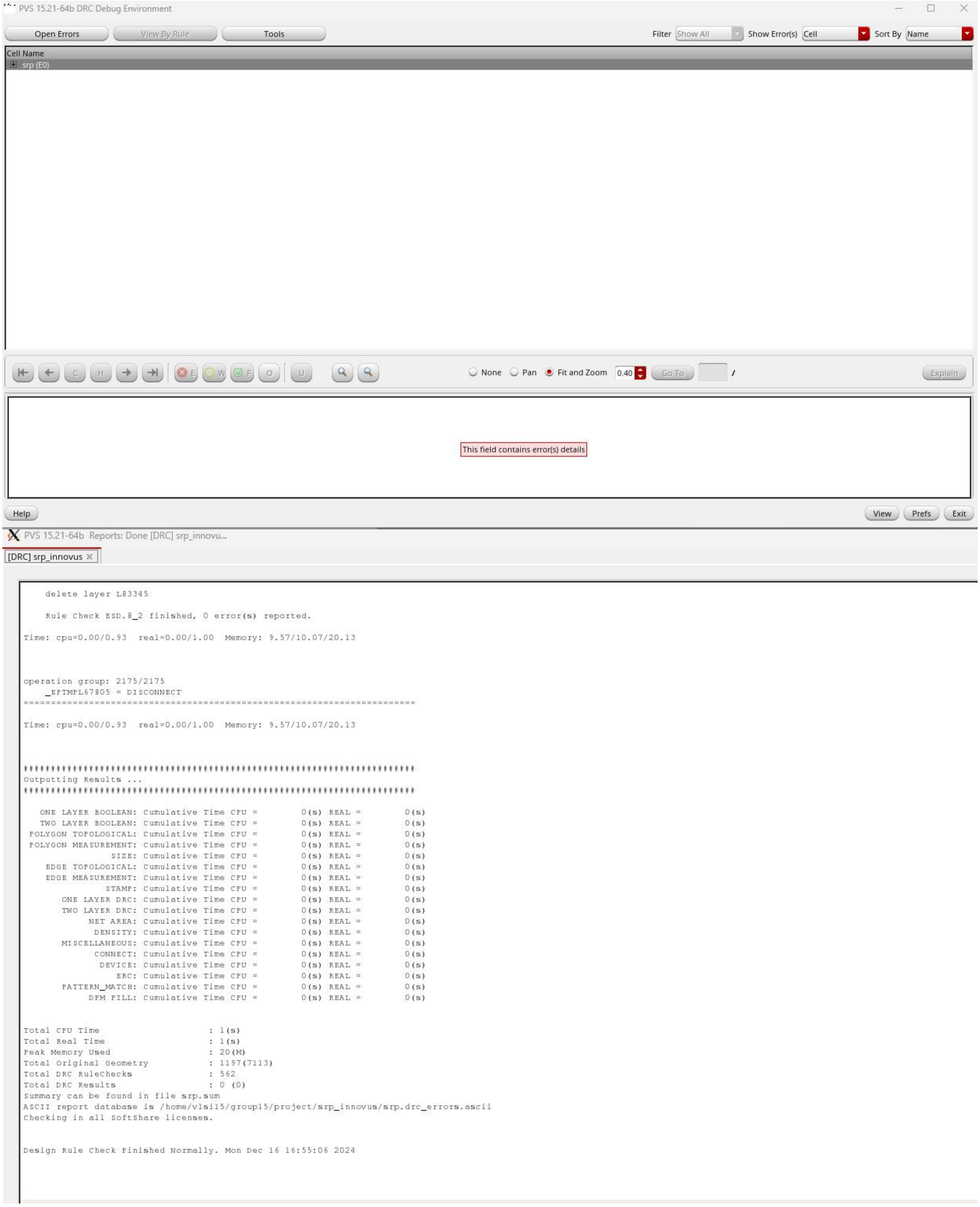

Open File
File
Library
srp
Cell
srp
View
layout
Type
layout
Browse
Application
Open with
Layout L
Always use this application for this type of file
Cells
AOI222X1
AOI22XL
DFFHQX1
FILL16
FILL2
FILL4
INVXL
M5_M4_VH
NOR2BX1
NOR2X1
OAI2BB1X1
srp
srp_VIA0
Open for
edit
read
Library path file
/home/vlsi15/cds.lib
OK
Cancel
Help



We have edited the dislay option to get the first figure. Then we removed labels by unticking them to get the following diagram.


Virtuoso® Layout Suite L Editing: srp srp layout
Launch File Edit View Create Verify Connectivity Options Tools Window QRC Optimize Help
cādence
Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 6.9400
Palette
Layers
All Layers
Valid Used Routing
Filter
Nwell drawing
AV NV AS NS
Layer Pu... V S
Nwell drw
Oxide drw
Oxide_thk drw
Poly drw
Pimp drw
Nhvt drw
Nlvt drw
Nimp drw
Phvt drw
Plvt drw
Nzvt drw
SiProt drw
Cont drw
Metal1 drw
Objects
Objects V S
Instances
Pins
Vias
Objects Grids
PreSel: Path Layer(Metal3 drw) Width(0.0800)
mouse L: mouseSingleSelectPt()
M: leHiEditDisplayOptions()
R: _lxHiMousePopUp()
1(2) >
Cmd:


Virtuoso® Layout Suite L Editing: srp srp layout
Launch File Edit View Create Verify Connectivity Options Tools Window QRC Optimize Help
cādence
Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 6.2500
Palette
Layers
All Layers
Valid Used Routing
Filter
Nwell drawing
AV NV AS NS
Layer Pu... V S
Poly drw
Pimp drw
Nhvt drw
Nlvt drw
Nimp drw
Phvt drw
Plvt drw
Nzvt drw
SiProt drw
Cont drw
Metal1 drw
Via1 drw
Metal2 drw
Via2 drw
Objects
Objects V S
Label
Shapes
Mosaic
Objects Grids
PreSel: Label Text(shift_reg[5]) Layer(Metal3 drw)
mouse L: mouseSingleSelectPt()
M: leHiEditDisplayOptions()
R: _lxHiMousePopUp()
1(2) >
Cmd:



Here we can see that DRC investigated for 562 cases and found zero violations. So our design is completely ready to be manufactured.

11 Reflection on Individual and Team work

Greatest part of our project group is that we worked as a team and we really helped each other for any errors or problems. All of us took their jobs seriously and we kept taking updates regularly. That really paid off finishing the work 1 week earlier.

11.1 Individual Contribution of Each Member

Name	ID	Contribution
Jarjis Mondal	1806068	Optimization and synthasis.
Jonaidul Islam Sikder	1906041	Physical design.
Md Mehedi Hasan	1906075	Layered testbench.
Md Faiyaz Abid	1906079	RTL code and directed testbench.

11.2 Log Book of Project Implementation

Date	Milestone achieved	Individual Role	Comments
05 October	Planning and workload distribution.	1906079, 1906075, 1906041, 1906068	Everyone got to know their responsibility for the project.
15 October	RTL code and directed testbench written	1906079	There were some errors and correction required.
15 November	Layered testbench written	1906075	Need to work more to get better coverage.

30 November	Optimization was completed. Synthesis was done with optimized time constraints. Area and power reports are generated.	1806068	Medium and high effort generates same report.
7 December	Physical design was done.	1906041	Die size needs to be optimized.

12 Conclusion

This project successfully demonstrated the design, implementation, and verification of a 16-bit Serial-In Serial-Out shift register. The design was implemented in Verilog and verified using both directed and layered verification methodologies to ensure comprehensive testing. The design was then optimized for timing constraints and synthesized using Cadence tools. The synthesized netlist was subsequently used for physical design and layout, followed by rigorous Design Rule Checks (DRCs) to ensure manufacturability. The project provided valuable experience in digital design, verification, and physical design methodologies, and highlights the importance of a structured approach in achieving a successful and optimized design.

13 References:

1. https://www.allaboutcircuits.com/textbook/digital/chpt-12/serial-in-serial-out-shift-register/
2. https://www.geeksforgeeks.org/shift-registers-in-digital-logic/
3. https://www.electro-tech-online.com/threads/16-bit-serial-in-parallel-out-shift-registers.33121/
4. https://verificationacademy.com/forums/t/layered-testbench/41565